U. S. N. A. – Trident Scholar project report; no. 273 (2000)

"NEURAL NETWORK CONTROL OF THE INTEGRATED POWER SYSTEM"

by

Midshipman Jonathan J. Cerrito, Class of 2000
United States Naval Academy
Annapolis, MD 21402

Certification of Advisor Approval

Assistant Professor Edwin L. Zivi
Department of Weapons and Systems Engineering

Lieutenant George D. Doney, USN
Department of Weapons and Systems Engineering

Acceptance of the Trident Scholar Committee

Professor Joyce E. Shade
Chair, Trident Scholar Committee

USNA-1531-2

REPORT DOCUMENTATION PAGE

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (FROM - TO)
07-05-2000	USNA Trident Scholar Project Report	xx-xx-2000 to xx-xx-2000

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Neural Network Control of the Integrated Power System	5b. GRANT NUMBER
Unclassified	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
Cerrito, Jonathan J. ;	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME AND ADDRESS	8. PERFORMING ORGANIZATION REPORT NUMBER
U.S. Naval Academy Annapolis , MD 21402	
9. SPONSORING/MONITORING AGENCY NAME AND ADDRESS	10. SPONSOR/MONITOR'S ACRONYM(S)
,	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

A

PUBLIC RELEASE

,

13. SUPPLEMENTARY NOTES
Accepted by the U.S. Trident Scholar Committee

14. ABSTRACT
Neural networks are investigated for fault tolerant stabilization and control of an Integrated Power System (IPS). Neural networks can be robust in the sense that they are not disabled by incomplete or inconsistent information. As non- model based observers, neural networks are ideally suited to estimation of complex, interactive power systems. Specifically, the ability of neural networks to adapt to uncertain eventualities such as flooding, fire, and combat casualties is investigated. The IPS under consideration will provide integrated propulsion and ship?s service power generation and distribution for the next generation of U.S. Navy surface ships also known as the DD-21. These solid state power systems involve nonlinear dynamics which can lead to ?negative impedance? instability and voltage collapse. Feedforward back-propagating neural networks were evaluated with respect to variable structure and data degradation. This research represents an initial step toward unifying nonlinear, negative impedance stabilization with robust neural network fault detection and isolation. The Naval Sea Systems, Integrated Power System and the Office of Naval Research, Electrically Reconfigurable Ship programs motivated this research.

15. SUBJECT TERMS
Neural Networks; Integrated Power System; Nonlinear Control Systems

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
					Fenster, Lynn lfenster@dtic.mil
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	Public Release	96	**19b. TELEPHONE NUMBER** **International Area Code** **Area Code Telephone Number** 703 737-9007 **DSN** 427-9007

Abstract

"Neural Network Control of an Integrated Power System"

Neural networks are investigated for fault tolerant stabilization and control of an Integrated Power System (IPS). Neural networks can be robust in the sense that they are not disabled by incomplete or inconsistent information. As non-model based observers, neural networks are ideally suited to estimation of complex, interactive power systems. Specifically, the ability of neural networks to adapt to uncertain eventualities such as flooding, fire, and combat casualties is investigated. The IPS under consideration will provide integrated propulsion and ship's service power generation and distribution for the next generation of U.S. Navy surface ships also known as the DD-21. These solid state power systems involve nonlinear dynamics which can lead to "negative impedance" instability and voltage collapse. Feedforward back-propagating neural networks were evaluated with respect to variable structure and data degradation. This research represents an initial step toward unifying nonlinear, negative impedance stabilization with robust neural network fault detection and isolation. The Naval Sea Systems, Integrated Power System and the Office of Naval Research, Electrically Reconfigurable Ship programs motivated this research.

Keywords

Neural Networks, Integrated Power System, Nonlinear Control Systems

Acknowledgments

LCDR Tim McCoy, IPS Technical Director, PMS 500, and LCDR Frank Novak, ONR 334, provided significant insight and background concerning the automation of advanced shipboard machinery. S.F. Glover, B.T. Kuhn, and S.D. Sudhoff of the Energy Systems Analysis Consortium developed the IPS simulation under ONR sponsorship. Jack Copper of NeuralWare provided no-cost access to the NeuralWorks Professional II/ Plus software.

Nomenclature

Backpropagation: a neural network learning characteristic in which the error is sent through the network from the output layer to the hidden layer(s) and then finally reaching the input layer.

Connection: the conduit supplying information from one neuron to the next neuron.

Epochs: the number of passes through a data file a neural network will make before updating the weights.

Fault Tolerance: ability of a system to continue to function when components of the system fail or are degraded.

Feedforward: a type of neural network in which the data is sent directly from the input to the respective hidden layers and then directly to the output layer.

Hyperbolic Tangent (TanH): a transfer function used to relate neuron input and output values in neural networks. It has an output range of –1 to 1. $T = \frac{(e^{I'} - e^{-I'})}{(e^{I'} + e^{-I'})}$, where I'=I*Gain.

Input: data entered into the neural network; parameters measured from the system.

Layer: a grouping of neurons with the same transfer function and learning rule.

Learning: the process by which a neural network adjusts its weights to model a relationship between the input and output data.

Neuron: the most basic element of a neural network. The learning rule and transfer function are applied directly to this component.

Output: data that is either produced by the neural network or is desired by the system.

Processing Element: see Neuron.

Robustness: ability of the system to deal with large changes in measured quantities and evolving topology or architecture of the system.

Sensitivity: capacity of a control system to determine minimal changes in measured quantities within the system.

Testing: the evaluation of how well a neural network has learned by supplying the network with input data and allowing it to predict outputs. These predicted outputs are then compared with the desired outputs to determine the accuracy of the neural network.

Tractable: ability to be solved in simplified terms.

Training: see learning.

Variable Structure: changes in a system's physical configuration.

Weights: factor applied to the connections between neurons in a network. It is determined through the learning process.

Table of Contents

1 Introduction

The effective management of electrical power will only increase in importance, as the navigation, weapons, and propulsion systems aboard naval ships grow more complex in nature. The need for efficient power management occurs during normal operations as well as casualty conditions such as flooding, fire, or enemy attack. This requirement presents the engineer with a unique challenge. The goal is to develop a system that is durable or robust and yet precise. This system must be quick to respond to voltage collapse. However, it must recognize routine changes in voltage as the ship carries on normal operations. Essentially, the system must be able to "think." Thanks to neural networks, a reasoning control system is entirely possible. [1,2,3,4]

The Naval Sea Systems Command, Integrated Power System (IPS) and the Office of Naval Research, Electrically Reconfigurable Ship (ERS) offer advanced power systems for deployment of advanced weapons, propulsion, and navigation systems on the next generation surface combatant. Because these systems often operate at constant power, they demand continuous regulation to prevent negative impedance instability.

Traditional control systems cannot perform these tasks due to the time constraints and the complexity of the control laws involved in managing the power system. Fortunately, neural networks can predict the states of the system faster than models can determine the states analytically. The complexity of the control laws demands a controller that has the flexibility and adaptability to manage a constantly changing environment. Neural networks possess this characteristic of robustness.

Neural networks will provide an IPS controller with the necessary robustness and speed. Their application to this advanced power grid will not only enable the use of state-of-the-art solid state components, but will assure the flexibility and survivability of a ship which must endure the harshest of all conditions, combat. [5]

This research explores the application of a neural network as a non-model based estimator of a shipboard power system. This power system must have the ability to continue to operate in all conditions including combat. In order to meet this demand, a controller for this power system must contend with multiple power plant configurations and corrupted data.
This investigation strives to resolve these issues through the design of neural networks. These networks are trained, tested, and evaluated on data that is derived from a reference model of the shipboard power system. The objective of this work is to provide a nonlinear, robust neural network as an alternative to traditional, linear controllers for the Integrated Power System.

2 Prior Work

2.1 Integrated Power System

The Secretary of the Navy has designated the next generation surface combatant, DD-21, as an Integrated Power System ship. The IPS is the next stage in an engineering design process that began in the 1970's with aero-derivative gas turbines. These were the first turbines in the Navy that demanded automated control. During the 1980's, the Navy procured the DDG-51 which featured Tactical Digital Standards (TADSTANDS) including Navy Standard Electronic Modules (SEMS) and the AN/USQ-82 Data Multiplexing System (DMS). Then, at the beginning of the 1990's, the Navy upgraded to a system based primarily on commercial standards with the Standard Monitoring and Control System (SMCS). SMCS included IEEE Futurebus+ computer backplane, IEEE Ethernet and ANSI Fiber Distributed Data Interface (FDDI) networks, and C software. Key goals for the DD-21 include a 70% crew size reduction and the ability to "fight thru" combat damage. [5]

To achieve these DD-21 goals, the Navy must develop an innovative control system. This control system must be robust, dynamic, survivable, and stable. Robustness involves the ability of the system to deal with large changes in measured quantities and evolving topology or architecture of the system. By comparison, sensitivity is the capacity by which a control system can determine small changes in measured quantities within the system. The control system must be dynamic, which is to say that it can change with time. It cannot be inflexible and unable to adapt to new conditions including the addition of new hardware and the loss of sensors. A key aspect of the neural network is its ability to distinguish between casualties and faulty data. Finally, stability or the ability to maintain a desired level of performance is a major concern in any control system.

2.2 Neural Networks

Simon Haykin states [6] "a neural network is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use." Essentially, it is an attempt by mathematicians to model the biological process that the brain conducts in order for humans to think.

The processing element of a neural network is equivalent to the human brain's neurons. Mathematicians have modeled the neuron through this equation:

$$y = j\left(\sum_{i=1}^{N} w_i x_i - b\right) \tag{2.1}$$

where x is the input, w is the weight, b is bias, and j is the transfer function, and y is the output of the neuron. The processing element (PE) or neuron accepts inputs from either other

neurons or directly from the input source. The input is processed by multiplying the weight and the input less the bias and summed over a series of N iterations. The transfer function then transforms this sum into an output. Other neurons may then accept this output as an input via connection weights that represent the strength of the connection within the network.

In this research, linear transfer functions were employed for the input and output layers while hyperbolic tangent functions were used for the hidden layers. Linear transfer functions described algebraic relationships among the inputs and outputs. The hidden layers were composed of hyperbolic tangent functions, which are commonly used for nonlinear applications. The middle layers are called hidden because they do not receive nor transmit input or output data.

Narendra and Parthasarathy [7] present an excellent discussion of the two fundamental theorems that support the mathematics behind neural networks. These theorems are the *Weierstrass Theorem* and the *Stone-Weierstrass Theorem*. The *Weierstrass Theorem* states that if $C([a,b])$ is a space of continuous real valued functions on the interval $[a,b]$ with the norm of $f \in C([a,b])$ defined by $\|f\| = \sup\{|f(t)| : t \in [a,b]\}$, then any functions in $C([a,b])$ can be approximated arbitrarily closely by a polynomial. This *Weierstrass Theorem* and its generalization to multiple dimensions are useful in approximating continuous functions $f : \mathbb{R}^n \to \mathbb{R}^m$ using polynomials. The *Weierstrass Theorem* is essentially the basis for pattern recognition. In addition, the *Stone-Weierstrass Theorem* is a generalization of the *Weierstrass Theorem* by Stone. This theorem states that [7]:

if b is a compact metric space and r is a subalgebra of $C(b,\mathbb{R})$ which contains the constant functions and separates points of b, then r is dense in $C(b,\mathbb{R})$.

The *Stone-Weierstrass Theorem* provides the basis for approximating bounded, continuous, time-invariant causal operators. This result forms the foundation for the neural network approximation of dynamic systems.

After the neural network has been constructed, the network must undergo a learning cycle in order to become productive. Learning is the process of modifying the weights to coincide with the correct output and input data. There are two significant types of learning, supervised and unsupervised. Supervised learning occurs when both input and output data are given to the network. Unsupervised learning entails only giving the neural network input data and letting the network determine the output on its own. This project will exclusively focus on supervised learning. Networks can also be referred to as hetero-associative or auto-associative. Hetero-associative networks are trained on data that have outputs that are different from the inputs. Auto-associative networks are trained on data in which both the input and output sets are identical. [8,9,10]

A learning rule typically governs the adjustment of the neurons. This project applies backpropagation methods via the Extended-Delta-Bar-Delta (EDBD) Learning Rule.

Backpropagation learning procedures assume that all weights are in error when an incorrect output is received. Consequently, the error is sent back through each layer while modifying the weights in each connection until the error reaches the input layer and the modification of weights ceases. The EDBD algorithm assigns a time-varying momentum rate, $m[k]$, and a time-varying learning rate, $a[k]$, to each connection in the network (k is time) in order to train the network and limit the error between predicted output and desired output. The variable learning rate and variable momentum rate yield:

$$\Delta w[k+1] = a[k]*d[k] + m[k]*\Delta w[k] \tag{2.2}$$

and

$$w[k+1] = w[k] + \Delta w[k+1] \tag{2.3}$$

where $w[k]$ and $\Delta w[k]$ are the connection weight and the connection delta weight respectively and time, k. $d[k]$ is the gradient component of the weight change at time k.

$$d[k] = \frac{\partial E[k]}{\partial w[k]} \tag{2.4}$$

where $E[k]$ is the value of the error at time k. In order to provide greater increases in areas of lesser slope than in areas of higher gradients, $\bar{d}[k]$ is a weighted, exponential average of the previous gradient components at time k:

$$\bar{d}[k] = 1 - q d[k] + q d[k+1] \tag{2.5}$$

where q is the convex weighting factor. Constants necessary to complete the EDBD algorithm are listed below:

k_a constant learning rate scale factor
k_m constant momentum rate scale factor
g_a constant learning rate exponential factor
g_m constant momentum rate exponential factor
j_a constant learning rate decrement factor
j_m constant momentum rate decrement factor
a_{max} upper bound on the learning rate
m_{max} upper bound on the momentum rate

The learning rate change, $\Delta a[k]$, for EDBD is:

$$\Delta a[\mathrm{k}] = \begin{cases} k_a^{-g_a(|\bar{d}[\mathrm{k}]|)}, \text{if } \bar{d}[\mathrm{k\text{-}1]}d[\mathrm{k}] > 0 \\ -j_a a[\mathrm{k}], \text{if } \bar{d}[\mathrm{k\text{-}1]}d[\mathrm{k}] < 0 \\ 0, \text{otherwise} \end{cases} \tag{2.6}$$

The momentum rate change, $\Delta m[\mathrm{k}]$, is:

$$\Delta m[\mathrm{k}] = \begin{cases} k_m^{-g_m(|\bar{d}[\mathrm{k}]|)}, \text{if } \bar{d}[\mathrm{k\text{-}1]}d[\mathrm{k}] > 0 \\ -j_m m[\mathrm{k}], \text{if } \bar{d}[\mathrm{k\text{-}1]}d[\mathrm{k}] < 0 \\ 0, \text{otherwise} \end{cases} \tag{2.7}$$

The learning rate and momentum rate are adjusted based upon the results from equations (2.6) and (2.7). In addition, $\bar{d}[\mathrm{k\text{-}1]}d[\mathrm{k}]$ determines whether the modification will be an increase or decrease. Finally, the conditions, $a[\mathrm{k}] \le a_{\max}$ and $m[\mathrm{k}] \le m_{\max}$, are imposed on this algorithm to limit excessive oscillations. These values are selected through trial and error to allow for accelerated convergence while preventing instability.

A neural network must complete training, recall, and evaluation stages of development in order to perform accurately. In the recall phase of development, the network is presented with an input and then is expected to produce the output. In feedforward neural networks, the information is passed in a direct manner from the input layer to the hidden layer or layers and finally to the output layer. Lastly, the network's output should be compared to the desired output in order to evaluate the validity of the training. If the network's error cannot be attributed to intrinsic noise, then more training is required. The completion of the training cycle can be determined both graphically and computationally. As the neural network continues to learn, its ability to match training set data will improve until the network begins to overfit the data. Memorization is evident on a plot when the neural network output matches stochastic data points rather than the underlying relationship. Numerically, the network is learning as long as the correlation is increasing for the independent test set and decreasing for the training set. When the correlation starts to decrease for the independent training set and increase for the training set, then the neural network is memorizing. Independent test data evaluates the network's ability to generalize. It is crucial during the recall phase to present the neural network with data that it has not previously processed in its training cycle in order to evaluate its performance on conditions to which it has not previously been exposed.

Neural networks are ideal for nonlinear robust controls because they possess two attributes. The first is pattern recognition. Neural networks can determine a relationship without the need of a first principle explanation to relate all of the variables. Essentially, neural networks apply the principle of induction while traditional controls fashion their relationships through the principle of deduction. Secondly, fault tolerance is an extraordinary characteristic of

neural networks. Traditional controllers fail when parts of their system are destroyed or disabled. If various components of a neural network including the sensors are destroyed or disabled, the network will continue to function. Although the network's performance will degrade as more faulted information is presented, it will not immediately fail, as most traditional controllers are prone to do. Consequently, neural networks are very applicable and practical control tools that have unexplored potential.

2.3 Model Structure

The Integrated Power System and proposed neural network based controller are distributed throughout the ship. These system components are modeled as distinct, interconnected, lumped parameter subsystems. The lumped parameter assumption models distributed elements using a finite number of ordinary differential equations. This formulation results in a coupled system of nonlinear, time varying differential equations. Depending on the complexity of the mathematical relationships, and the number of variables in the system, the representation of system state can become very involved and difficult to solve. The following four categories of state equations range from general, difficult to solve formulations to simplified linear models with straightforward solutions:

- Implicit, Non-linear, Time-Varying:

$$f(\dot{\mathbf{x}}(t), \mathbf{x}(t), \mathbf{u}(t), t) = 0 \tag{2.8}$$

- Explicit, Non-linear, Time Varying:

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), t) \tag{2.9}$$

- Explicit, Linear, Time Varying:

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t) \tag{2.10}$$

- Explicit, Linear, Time Invariant:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \tag{2.11}$$

In a lumped system, the system can be described by a finite number of state variables using one of the equations in (2.8) through (2.11). The model of the system depends upon state space equations plus an output equation of the form:

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \tag{2.12}$$

where **u** is the input vector, **x** is the state variable vector, and **y** is the output vector. The **C** and **D** matrices map the state and input vectors to the output vector.

In nonlinear control theory, researchers have made certain assumptions in order to make nonlinear systems tractable or to simplify the system from a state equation such as the one presented in equation (2.8) to an equation like the one presented in (2.11). In equation (2.8), the relationships between the system state, **x**(t), the derivatives of the system state, $\dot{\mathbf{x}}(t)$, and the exogenous input, **u**(t), are implicit, nonlinear, and time varying. Equation (2.9) describes an explicit formulation for $\dot{\mathbf{x}}(t)$ obtained through separation of algebraic and differential portions of equations. The Integrated Power System Simulation, employed in this study, uses the equation (2.9) formulation. This model is a variable structure, in the sense that the system configuration changes with time. Equation (2.10) represents a state space model in terms of matrices **A**(t) and **B**(t) which are linear, but time varying. Equation (2.11) represents the simplest of state space models, an explicit, linear, time invariant system. For convenience, the preliminary investigation was performed using a simplified electric circuit in the form of equations (2.10) and (2.11).

The attraction to simplify an implicit, nonlinear, time varying system to an explicit, linear, time invariant (LTI) system arises from the standard procedures common only to linear systems for determining controllability, observability, and stability. Although these assumptions were successful in solving some nonlinear systems, they cannot be applied to all instances because of the complexity and unpredictability of nonlinear problems. However, a neural network can and has been applied to explicit, nonlinear, time varying systems or those similar to equation (2.9) and been successful. [7]

2.4 Prior Research

Work in applying neural networks in practical control systems has been ongoing for many years. In 1990, Roger Barron et al. presented a paper [11] at the National Aerospace Electronics Conference. The foci of this paper are Fault, Detection, Isolation, and Estimation (FDIE) functions and Reconfigurable Flight Control. FDIE functions are challenged by the inability to maintain complete observability during a multitude of casualty situations in an aircraft. This large number of casualty situations presents many potential operational fault conditions. In addition, the control issues involved in casualty cases are high order, time-varying and nonlinear. The control processes have multiple variables and cannot be fully observed. Finally, there are undefined aero-inertial parameters and undefined aeroservoelastic characteristics especially in aircraft which are damaged. A controller designed to handle these issues must be able to adapt to real time conditions and learn to deal with unforeseen conditions.

Reconfigurable Flight Control demands adherence to four principles of reconfigurable control law. First, estimated effector sensitivities should be used to adjust control law gains

via an explicit pseudo-inverse calculation. Secondly, estimated effector sensitivities should be used to adjust control law gains via an implicit or neural network calculation. Third, fixed and implicit (polynomial neural network) calculations should be used to infer control law gains independently of estimated effector sensitivities. Finally, implicit, on-line adaptive (polynomial neural network) calculations should be used to command the control effectors. Barron et al. derived the Algorithm for Synthesis of Polynomial Networks – II (ASPN-II) for feedforward polynomial neural network in a supervised, offline environment. These networks are composed of a linear combination of polynomial transfer functions. An implicit model of data containing a maximum of 200 input and output pairs was used to train the neural network online. Barron et al. concluded that polynomial neural networks are potential solutions to FDIE problems in reconfigurable flight systems since they responded by scoring a 94.4% probability of detecting and correctly isolating effector impairments exceeding 50% missing. [11]

Roger Barron and Eugene Parker wrote a white paper [4] in 1993 concerning the applicability of neural networks to smart shipboard systems. Barron and Parker address prediction, FDIE, and reconfiguration using neural networks in smart shipboard systems. Barron et al. compare neural network controllers to expert system controllers. Hard thresholds do not limit neural networks in making decisions. Neural networks can reduce the number of rules inherent in expert systems. Neural networks utilize all pertinent observable data including parameter that may be unfamiliar to humans. Neural networks can also receive data from various sources and organize it into a coherent control strategy. Neural networks are faster than expert systems in gathering input data. Neural networks can also predict future states of instability and recognize potential casualties.

Barron et al. continue in their discussion to address fault detection and isolation (FDI) in Shipboard Electric Power Distribution Systems. Barron's polynomial neural network was used to control an SPD Technologies solid-state circuit breaker that can open high-current power circuits within a few microseconds. This circuit breaker was able to protect power system components when certain fault classes instigated ten ampere/second current growth rates. In this application, a polynomial neural network relying on its pattern recognition capability dramatically improved fault detection and identification. This system has been experimentally validated for three-phase AC power protection under normal, bolted-fault, and arcing-fault conditions. [4]

In 1995, Guglielmi et al. applied [3] neural networks to solving real fault detection and diagnostic problems in four heaters of a feedwater high-pressure line of a 320 MW power plant. Their results show that neural networks can function in such an environment. The neural networks used by Guglielmi et al. in this investigation were able to adequately diagnose the system with respect to steady-state operation. Additionally, in transient modes of operation, one of their neural network designs performed very well in detecting and diagnosing faults.

Liu, Su, Tsay, and Wang [1] investigated the application of measuring phasors in order to predict better real-time transient stability swings. They used specific Phasor Measurement Units (PMUs) to find the phasor measurements throughout the system. Liu et al. showed that neural networks can predict the behavior of a system faster than the behavior can be computed analytically. Since they used an eight cycle window of phasor measurements, the neural network was able to select the most pertinent data from an overdetermined set. They used a combination of Supervised Decision Directed Learning and Backpropagation to train their network. Liu et al. also used counterexamples of data to prevent the neural network from "memorizing" the relationships evident within the system.

3 Technical Approach

The advanced Integration Power System is essential to the design of DD-21, the next generation surface combatant. Without this power system, the ship will not be able to employ the most sophisticated and state-of-the-art weapons, navigation, and propulsion systems. These advanced systems and their solid-state components demonstrate non-linear behavior with respect to power consumption. At the moment, no control system can effectively manage these components as well as complete the necessary tasks in the event of a casualty. It is the objective of this project to develop a control system through the employment of feed-forward back propagating neural networks that will not only manage the advanced Integrated Power System, but also be ready to contend with the unexpected casualty.

There are two fundamental issues that a controller must address in order to operate proficiently within the IPS environment. The first of these issues is the variable structure of the power system. As the ship conducts operations, the configuration of the electrical system and the electrical system's loads will change. This dynamic environment presents a unique challenge to a controller since there are multiple scenarios that it must consider. The second fundamental issue is the fault tolerance of the controller. The controller must be able to contend with both a complete sensor failure due to a casualty and degradation in the sensor data. A competent controller will be able to cope with these two fundamental issues inherent to the Integrated Power System.

4 Preliminary Investigation

4.1 Simulation Description

As an important first step toward the ultimate goal of controlling an Integrated Power System, preliminary research was conducted on the simple circuit presented in figure 4-1.

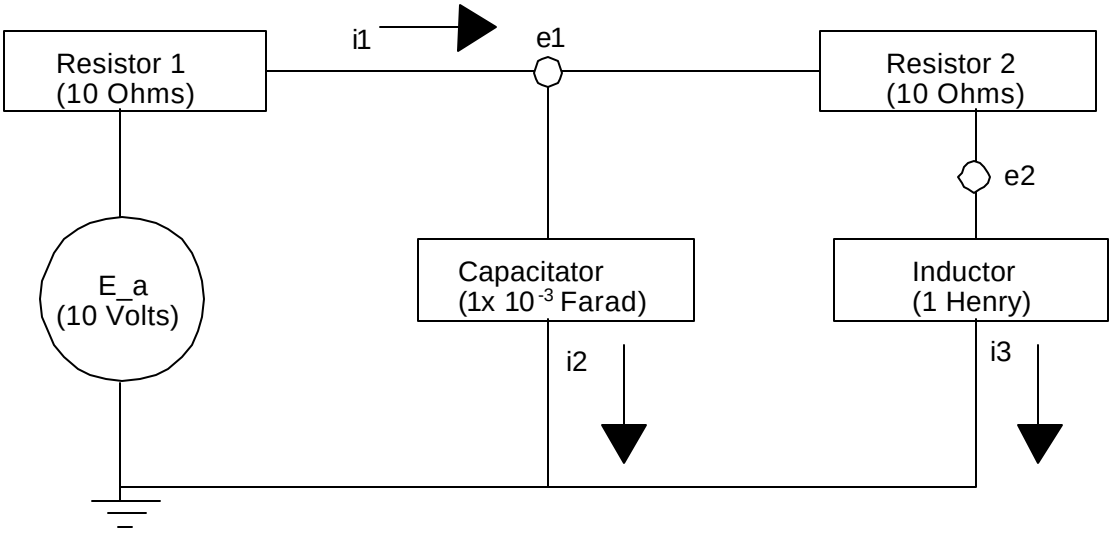


Figure 4-1 Simple Circuit

This simple circuit represents a second order initial value problem formulated in the state space equation (4.1). This circuit includes two resistors, a capacitor, an inductor, and a power source.

$$\frac{d}{dt}\begin{bmatrix} e_1 \\ i_3 \end{bmatrix} = \begin{bmatrix} \frac{-1}{R_1 C_1} & \frac{-1}{C_1} \\ \frac{1}{L_1} & \frac{-R_2}{L_1} \end{bmatrix} \begin{bmatrix} e_1 \\ i_3 \end{bmatrix} + \begin{bmatrix} \frac{1}{R_1 C_1} \\ 0 \end{bmatrix} E_a \qquad (4.1)$$

$$\begin{bmatrix} e_1 \\ i_3 \\ i_2 \\ e_2 \\ i_1 \end{bmatrix} = \begin{bmatrix} E_a & 0 \\ 0 & E_a \\ \frac{-E_a}{R_1} & -E_a \\ E_a & -E_a R_2 \\ \frac{-E_a}{R_1} & 0 \end{bmatrix} \begin{bmatrix} e_1 \\ i_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{1}{R_1} \\ 0 \\ \frac{1}{R_1} \end{bmatrix} E_a \tag{4.2}$$

After developing the state-space equations necessary for this circuit, a MATLAB simulation was used to determine the values of three currents and two voltages. To allow the neural network to estimate derivatives, each input record contained the current data augmented with shifted data. Effectively, each input record contained present data and one time step previous data. This z^{-1} time shift operation was performed in MATLAB:

$$\mathbf{Y}_{aug} = [\mathbf{Y} \vdots z^{-1}\mathbf{Y}] \tag{4.3}$$

or equivalently:

$$y_{aug}(n) = [y(n) \vdots y(n+k)] \tag{4.4}$$

Equations (4.3) and (4.4) can be interpreted as augmenting the system response time series with the system response time series shifted by Δt.

MATLAB was used to generate the preliminary simulation results and to augment the results with time shifted data. The MATLAB `step` command was used to determine the unit step response given equations (4.1), (4.2), and initial conditions of zero. To generalize the results, the MATLAB `lsim` command was used to determine the response given arbitrary initial conditions. The generalized, `lsim` based, simulation script and the MATLAB script to augment the simulation time series are enclosed in Appendix A.

4.2 Transition from MATLAB to NeuralWare

Although the initial neural network research was conducted using the MATLAB Neural Network Toolbox, it was necessary to change neural network environments in order to develop a more refined and flexible neural network.

MATLAB's Neural Network Toolbox is limited in its ability to construct neural networks for this project:

- Absence of visualization capabilities
- Inability to connect and disconnect specific weights within the neural network
- Primitive mechanism for specifying the structure of the neural network

Due to these limitations in MATLAB's Neural Network Toolbox, it became necessary to adopt the NeuralWare Professional II/Plus software.

NeuralWare is a more flexible program since it does allow for a connection between individual neurons in the input layer and individual neurons in the hidden layer. This attribute of NeuralWare greatly enhanced the capability of the neural network. NeuralWare also allows for the adjustment of each neuron's transfer function and learning rule. Thus, the ease with which one may operate NeuralWare as well as the visual representation available in NeuralWare make it a superior tool for developing neural networks when compared to MATLAB.

4.3 Neural Network Design

The neural network for this preliminary investigation is a feedforward backpropagating neural network. Figure 4-2 presents the overall architecture of the neural network. It has seven total layers: one input layer, one output layer, and five hidden layers. The input layer has ten processing elements or neurons in order to accept the five selected parameters of i_1, i_2, i_3, e_1 and e_2 and the five time-shifted parameters of i_1, i_2, i_3, e_1 and e_2. These ten processing elements used a linear transfer function and did not apply a learning rule. A learning rule was not necessary for these neurons since their input did not pass through a previous neuron. In other words, the data came directly from the sensors.

The input layer neurons were then connected via weighted connections to the five hidden layers that applied the hyperbolic tangent transfer function and the EDBD learning rule. All input neurons, except the neurons that contained the data to be estimated, were connected to the hidden layers. This configuration was chosen to prohibit memorization and force the neural network to learn the relationship between the other eight parameters. Each hidden layer contained three processing elements. Thus, the hidden layer responsible for estimating i_1 did not receive input from i_1.

The three hidden layer neurons were then connected directly to the output layer neuron that was responsible for producing the parameter absent from the hidden layer. The output layer connections also used a linear transfer function and the EDBD learning rule. There were also connections that bypassed the hidden layer and directly connected input neurons to the appropriate output neurons. Finally, a bias was connected to the hidden layer and output neurons.

Since neural networks have no intrinsic ability to model dynamic systems, it was necessary to augment the input with time-shifted data. These time-shifted data provided the neural network with a way to estimate derivatives using finite differences. Entering the time-shifted data is necessary because the neural network is “memoryless”. Better results could have been obtained by explicitly providing the neural network derivatives. Time shifted data was preferred over finite differences due to ease of implementation in real world scenarios.

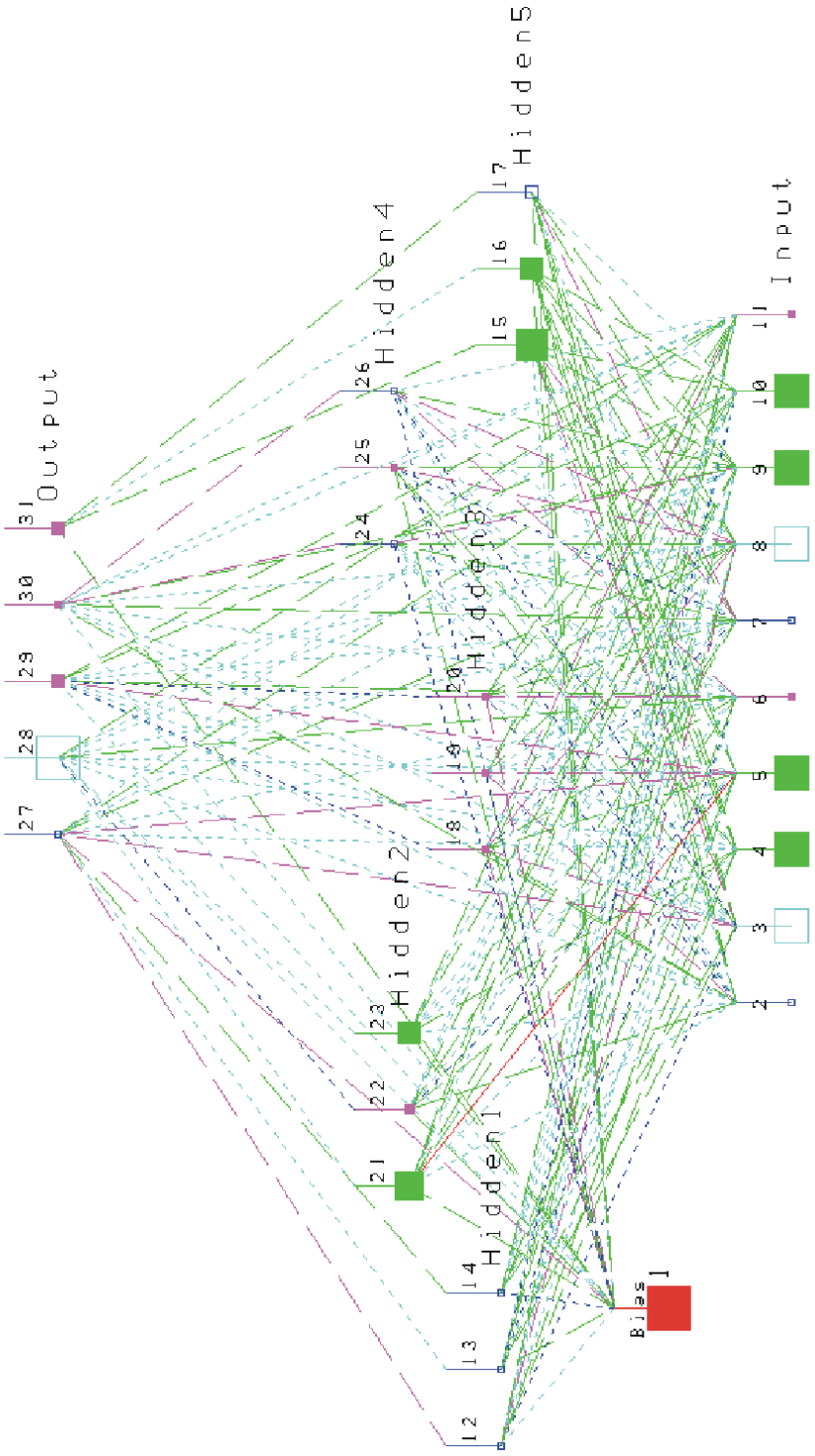


Figure 4-2 Preliminary Neural Network Architecture

4.4 Training Data

The neural network was trained on three data sets. Initial training was conducted on sequential, individual data sets that had completely accurate data. This training method did not sufficiently train the neural network because it over emphasized the most recent training set. To alleviate this problem, a single set of training data was constructed from individual simulation results. This operation was performed by concatenating the individual training sets into a single, aggregated training set. This change in training methods allowed the neural network to weigh all training data equally. The neural network is unaware of the sequential nature of the data. It randomly accesses the individual observations and each observation represents a strobe of system state at a particular instant of time.

To achieve a fault tolerant neural network, the training set was modified to include sample sensor faults. The sample faults included random variations and "drop outs." Random variations of $\pm 10\%$ were imposed to simulate sensors that were degraded or out of calibration. "Drop outs" are a complete loss of signal, representing a loss of sensing capability due to sensor failure of battle damage. In essence, this procedure trained the neural network to be robust with respect to degraded sensor data. The quantity of faulty data was limited to approximately 15% to minimize the adverse effects on network performance metrics. During training, minimum summed squared error and other performance metrics guide the neural network to the generalized solution. Increasing the proportion of faulty data would drive the generalized solution to represent fault conditions rather than normal conditions.

The amount of faulted data in comparison to the accurate data was also a concern during training. The percentage of faulted data should not be greater than the percentage of accurate data. In reality, the percentage of faulted data should be between 10% and 30%. The accurate data should also not be derived from the same initial conditions. The neural network's accuracy was greatly enhanced when varying initial conditions were used to produce accurate training data.

At this point, the neural network was able to reproduce the transient response but was not a good predictor of initial conditions. This poor performance is attributable to the fact that initial conditions represented only one out of 250 observations. To achieve a balanced weighting of initial and transient response, the proportion of observations that represents initial, steady state data was increased. This balancing was performed by extruding the initial conditions in time. This extrusion repeated steady state conditions over an initial period of time to increase the weight associated with initial conditions.

4.5 Testing & Results

After training the neural network, it was necessary to evaluate its performance.

The testing of the data was carried out by producing three sets of original test data. The neural network was not exposed to this data at any point in its training cycle. By keeping the test data separate from training data, the neural network was not given the opportunity to recite memorized data. Instead, this separation allowed for the critical evaluation of the relationship that the neural network had established to model the system. Thus, the test set was completely foreign to the neural network and provided an accurate measurement of the neural network's performance.

The neural network's best performance was produced after having trained for 10,000 epochs on a combined data set of 10,500 points including varying initial conditions, data drop outs, and ten percent error data. One epoch represents a single pass through the training data set. As shown below, the results of testing that included missing data were very good.

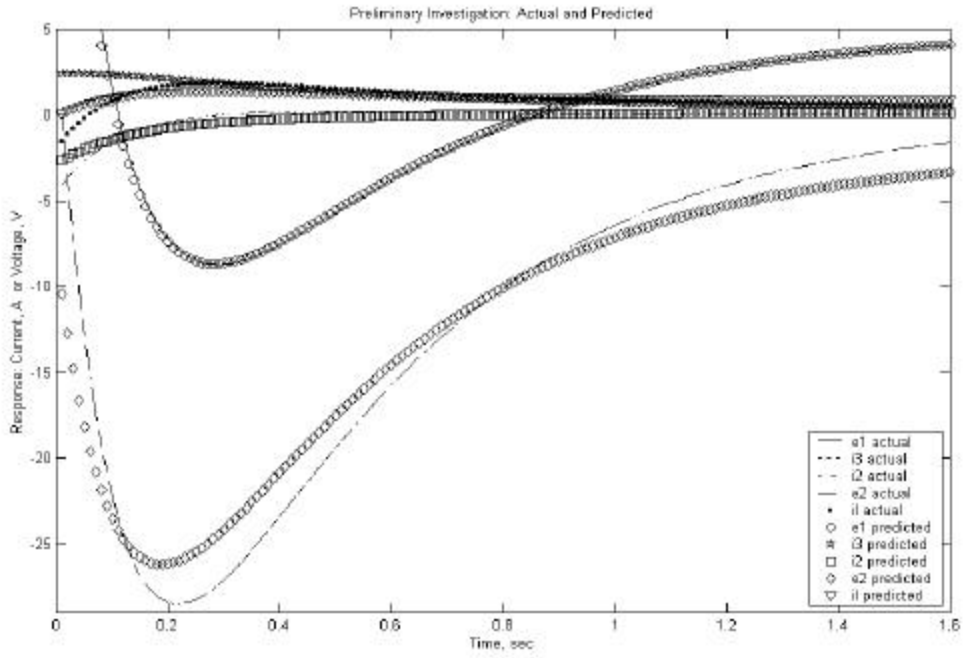


Figure 4-3 Preliminary Neural Network Response to Missing Data

In this plot, the parameter, e1 and e1's time shifted data were replaced with all zeros to simulate a sensor failure. In this scenario, the network was able to compensate accurately for the lack of e1 data with respect to predicting e1. However, since the e1 data have a strong relationship to e2, the neural network was not able to predict e2 exactly. The zero columns of e1 and e2 have a much more dramatic effect on the other parameters in this neural network since there are only 10 total parameters. Having considered this effect, the results shown in this plot are even more impressive.

Comparatively, when the network was tested on data that had a ten percent input error, the neural network performed well again. In Figure 4-3, one can see that every 40 data points, there is a slight shift in the network's prediction. Due to the alternating between parameters of the missing data, the network is able to maintain accurate predictions and handle a sensor

fault in any parameter. Additional sensors would further mitigate the adverse effect of a loss of sensors.

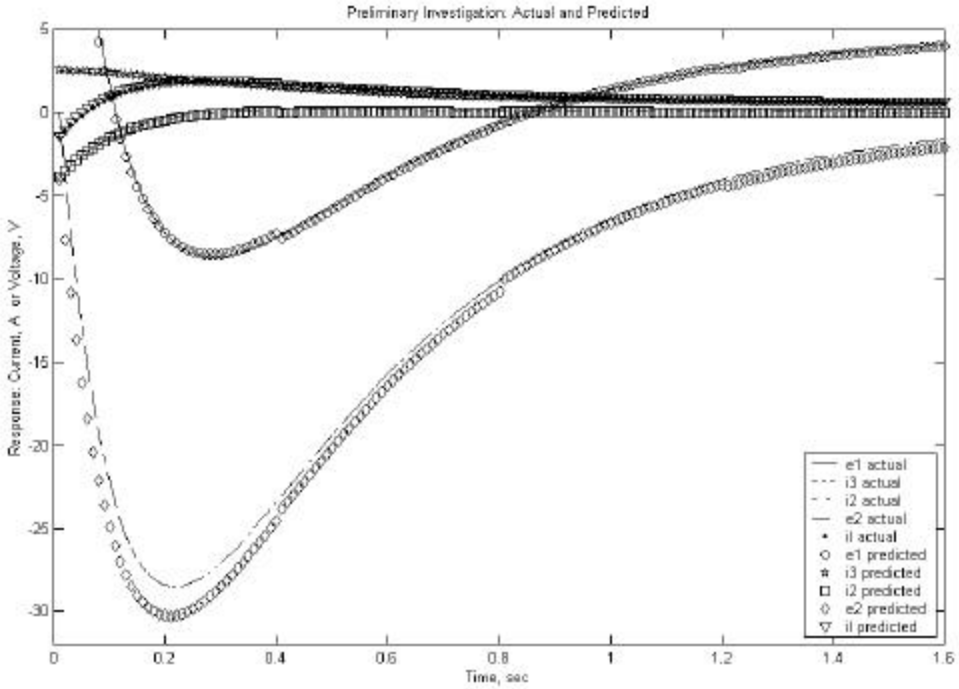


Figure 4-4 Preliminary Neural Network Response to Degraded Data

These positive results paved the way for investigation of the Integrated Power System.

5 Investigation of IPS Reference Model

5.1 Description of the Integrated Power System Simulation

The Integrated Power System DC Zonal Electrical Distribution System or IPS DC ZEDS is a vital component of the DD-21, next generation surface combatant project. This power system provides the necessary power generation, distribution, and conditioning to support the advanced weapons, navigation, and operating systems required by the mandates for DD-21.

The IPS simulation used in this project is a subset of the entire IPS system. It is complete in the sense that it models the necessary components, connections, and architecture that are present in the total system. S. F. Glover, B. T. Kuhn, and S. D. Sudhoff of Purdue University developed this IPS simulation under Office of Naval Research funding. [12]

The Naval Sea System Integrated Power System (IPS) and Office of Naval Research Electrically Reconfigurable Ship (ERS) provide advanced power distribution systems that can reroute power around damage and maintain power continuity to vital loads.

As shown in Figure 5-1, these solid state power electronic modules include [12]:

- Power generation module composed of a Prime Mover (PM), Synchronous Machine (SM), and voltage regulator
- Power Conversion Modules (PCMs)
- Ships Service Converter Modules (SSCMs)
- Propulsion Motor Module (PMM).

All of these components are interconnected for improved flexibility and survivability.

A 19 mega-Watt synchronous generator powers the IPS reference model. This 4160 Volt AC Bus 1 distributes power to propulsion and redundant ships service supply busses. Electric propulsion power is converted to mechanical power via a 19 mega-Watt induction motor and drive system. Simultaneously, the Ship Service Power Supply (SSPS) converts AC Bus 1 power to 1100V DC for port and starboard distribution. Port and starboard Ship Service Converter Modules (SSCMs) convert the 1100V distribution power to DC to 900V and 860V DC. Auctioneering diodes draw power from whichever bus has the highest voltage potential. This feature allows bumpless transfer from the primary to the secondary supply bus. AC loads are served via Ship Service Converter Modules (SSCMs), producing 440V AC power. Finally, large DC loads are directly connected to the port and starboard DC busses via auctioneering diodes.

Glover et al. modeled this system through the use of the Advanced Continuous Simulation Language (ACSL). It is a computer simulation that accurately represents the behavior of the system. First, system components were modeled through a series of differential equations. These equations were then combined to produce a mathematical model of the system. Glover et al. then wrote a computer simulation program in the ACSL language based on this mathematical model. [12]

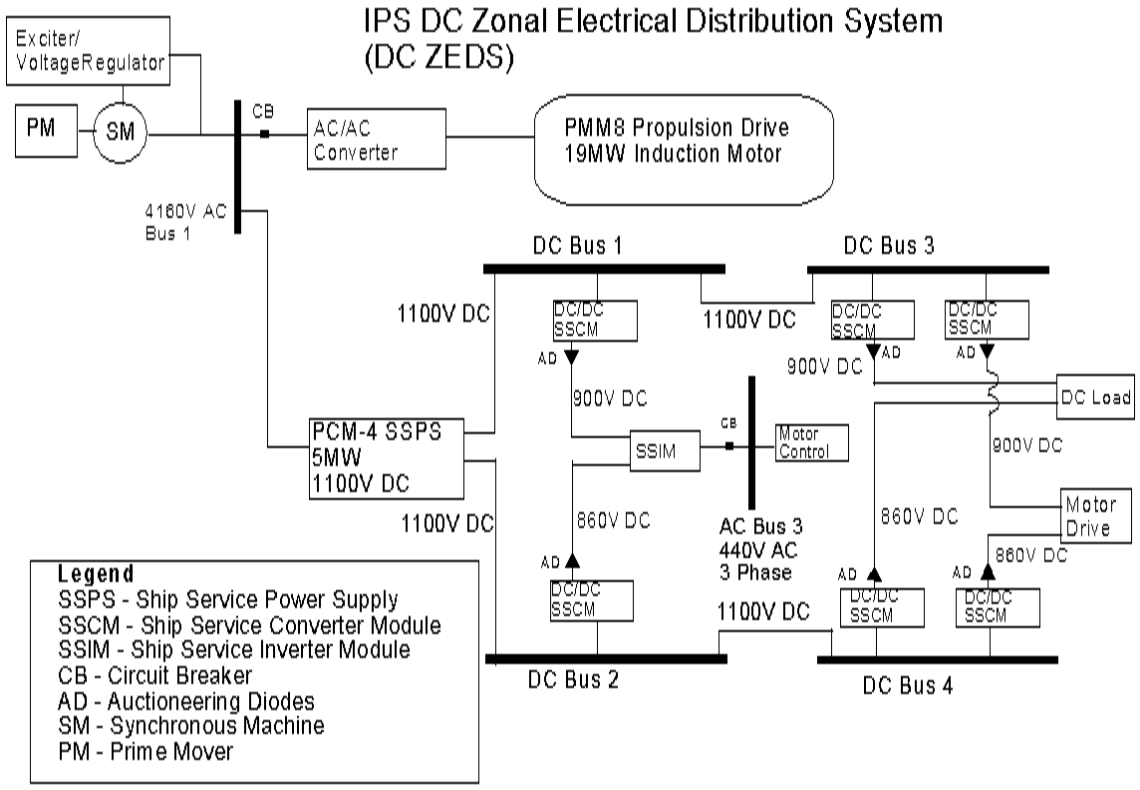


Figure 5-1 IPS Reference Model

5.2 IPS Simulations

The Integrated Power System reference model, developed by the Energy Systems Analysis Consortium (ESAC) was modified for this investigation [13]. All AC three phase voltages and currents were converted to RMS magnitudes. Run time scripts were composed to configure the system in eight distinct operating alignments. Once steady state operation was achieved, each run consisted of a single transient event. This single event was the activation of the induction motor attached to DC Buses 3 and 4 (commonly referred to as the "motor"). These runs of different configurations lasted 10 seconds and had a sample rate of 100 Hz. A total of eight runs were completed and logged. The initial settings for each run included energizing the Exciter/Voltage Regulator, the Prime Mover, the Synchronous Machine, AC Bus 1, and the AC to DC Converter, while connecting DC Bus 1 to DC Bus 3 and DC Bus 2 to DC Bus 4. The Propulsion Drive remained off. These two components were disconnected because their reaction to change in the power system was at least an order of magnitude slower. Consequently, their behavior would have minimal effect on the first 10 seconds of the motor start. While each run deviated from these initial settings in its configuration, the

same twenty parameters were observed during each run. The following table names these parameters and briefly describes them:

Table 5-1 IPS ACSL Simulation Variable Names

Voltages	Description	Currents	Description
vmagb1	AC Bus 1 Voltage	imagserv	Ships Service Load Current
vmagb3	AC Bus 3 Voltage	imagb3	AC Bus 3 Current
Vbus1	DC Bus 1 Voltage	ibus1	DC Bus 1 Current
Vbus2	DC Bus 2 Voltage	ibus2	DC Bus 2 Current
Vbus3	DC Bus 3 Voltage	ibus3	DC Bus 3 Current
Vbus4	DC Bus 4 Voltage	ibus4	DC Bus 4 Current
voutsscm3b	Voltage from DC Bus 3 to Diode Bridge	ibus1load	DC Bus 1 Load Current
voutsscm4b	Voltage from DC Bus 4 to Diode Bridge	ibus2load	DC Bus 2 Load Current
Vdc34b	Voltage out of Auctioneering Diodes	ioutsscm3b	Current from Bus 3 to Diode Bridge
		ioutsscm4b	Current from Bus 4 to Diode Bridge
		idc34b	Current out of Auctioneering Diodes

5.3 Configurations

This section deals exclusively with the configurations used to train and test the neural networks. Quantitative and qualitative results are presented in subsequent chapters.

5.3.1 Initial Configuration

Prior to commencing any simulations, all power components were disconnected. However, the DC Buses remained connected. Then, before every run, each component was energized from this initial configuration.

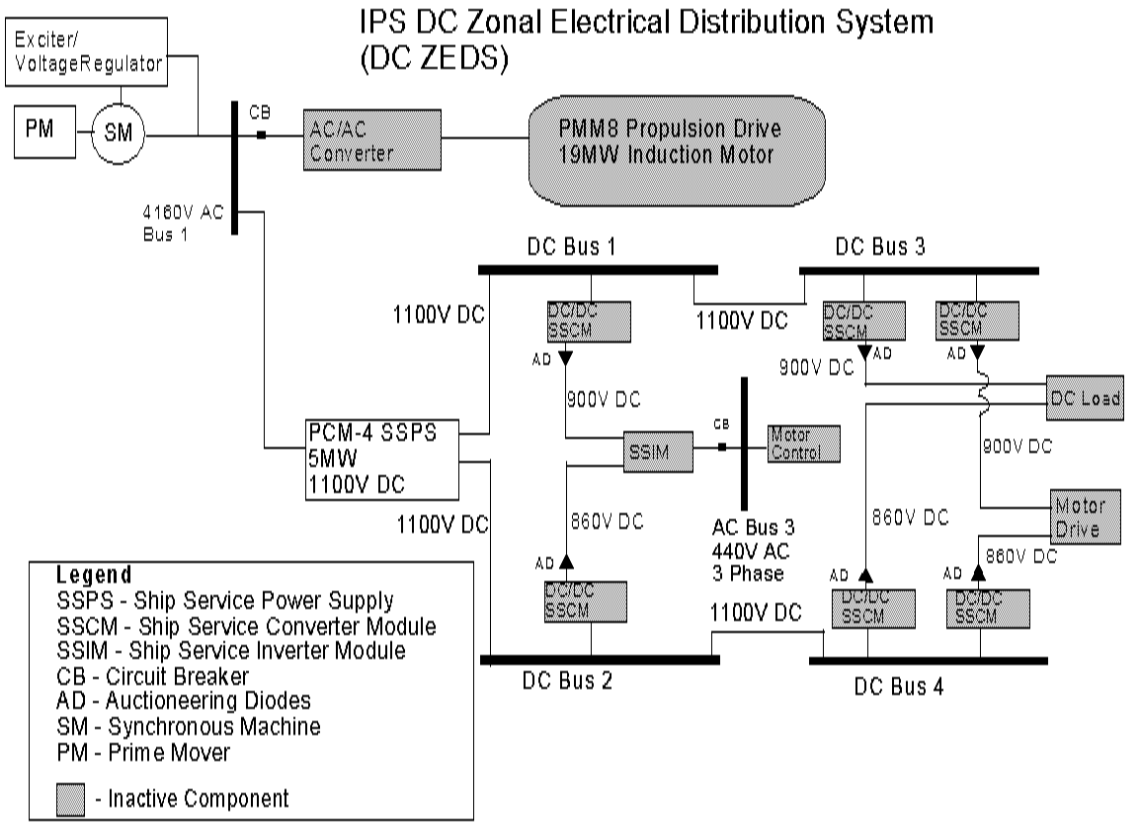


Figure 5-2 Initial Configuration

5.3.2 Run 1 Configuration

Run 1's configuration included an energized DC Bus 1 and DC Bus 3 with an active DC load on DC Bus 3. DC Buses 2 and 4 were disconnected.

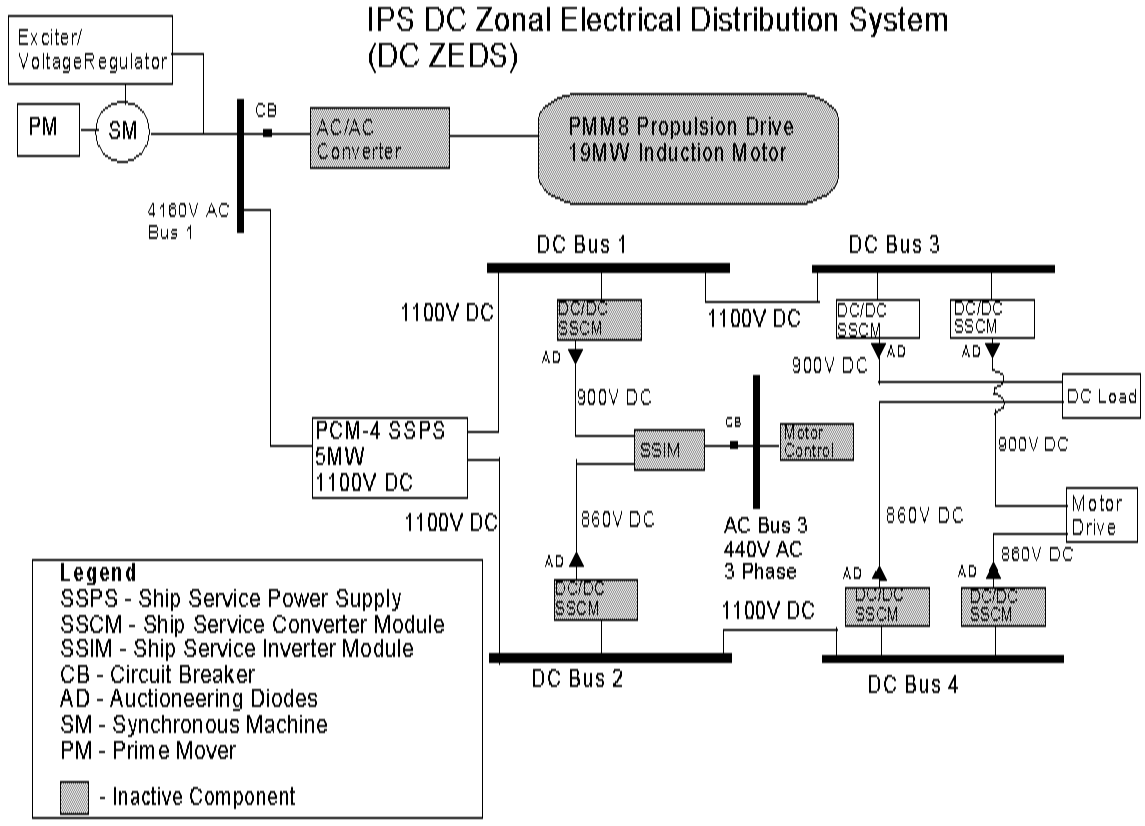


Figure 5-3 Run 1 Configuration

5.3.3 Run 2 Configuration

In the configuration for run 2, DC Buses 1 and 3 were disconnected while DC Buses 2 and 4 remained active with a resistive DC load on DC Bus 4.

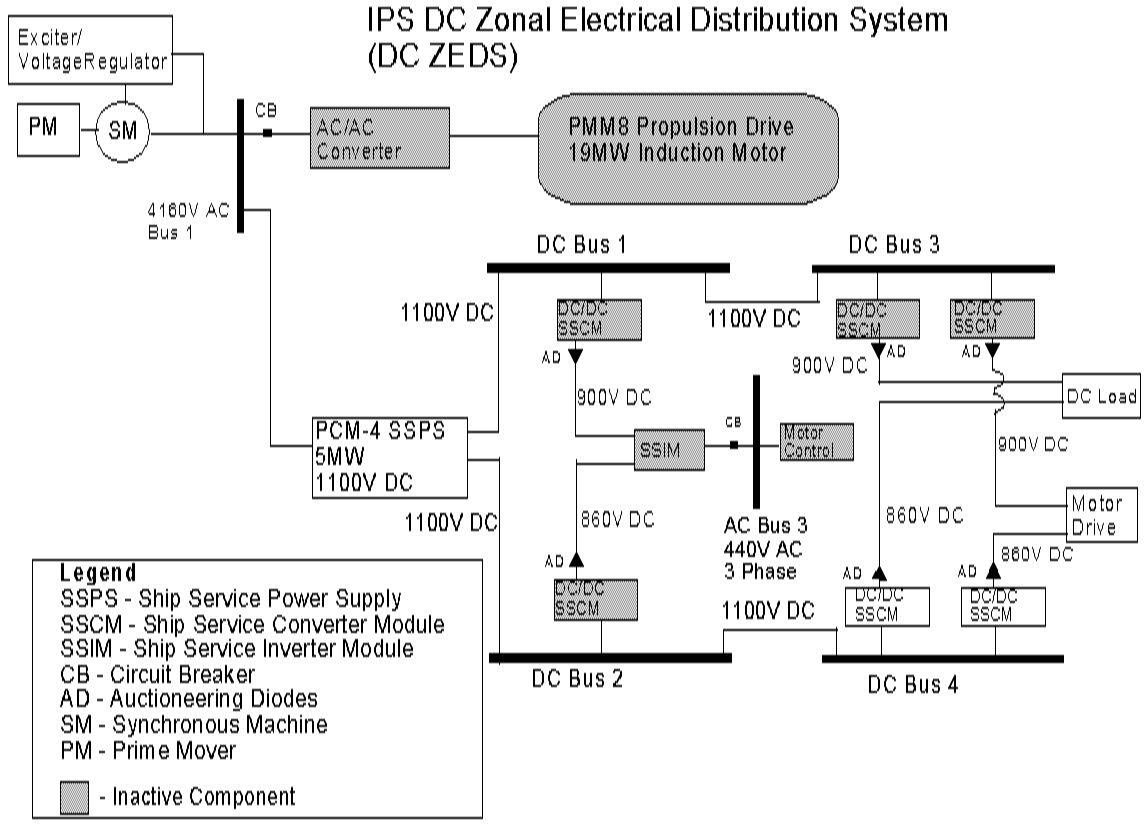


Figure 5-4 Run 2 Configuration

5.3.4 Run 3 Configuration

Run 3 was similar to Run 2, but it added the Ships Service Inverter Module to DC Bus 2 that energized AC Bus 3.

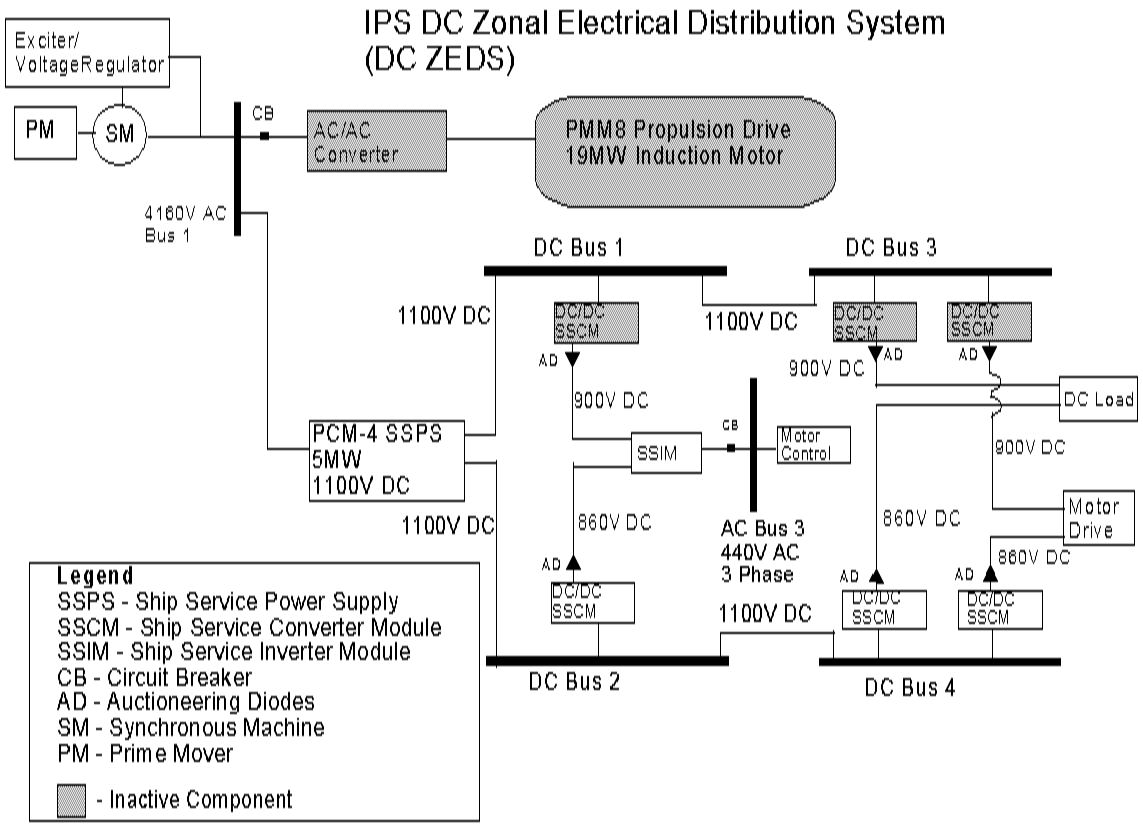


Figure 5-5 Run 3 Configuration

5.3.5 Run 4 Configuration

Run 4 built on the Run 3 configuration by adding DC Bus 1 to the Ships Service Inverter Module. However, DC Bus 4 remained connected.

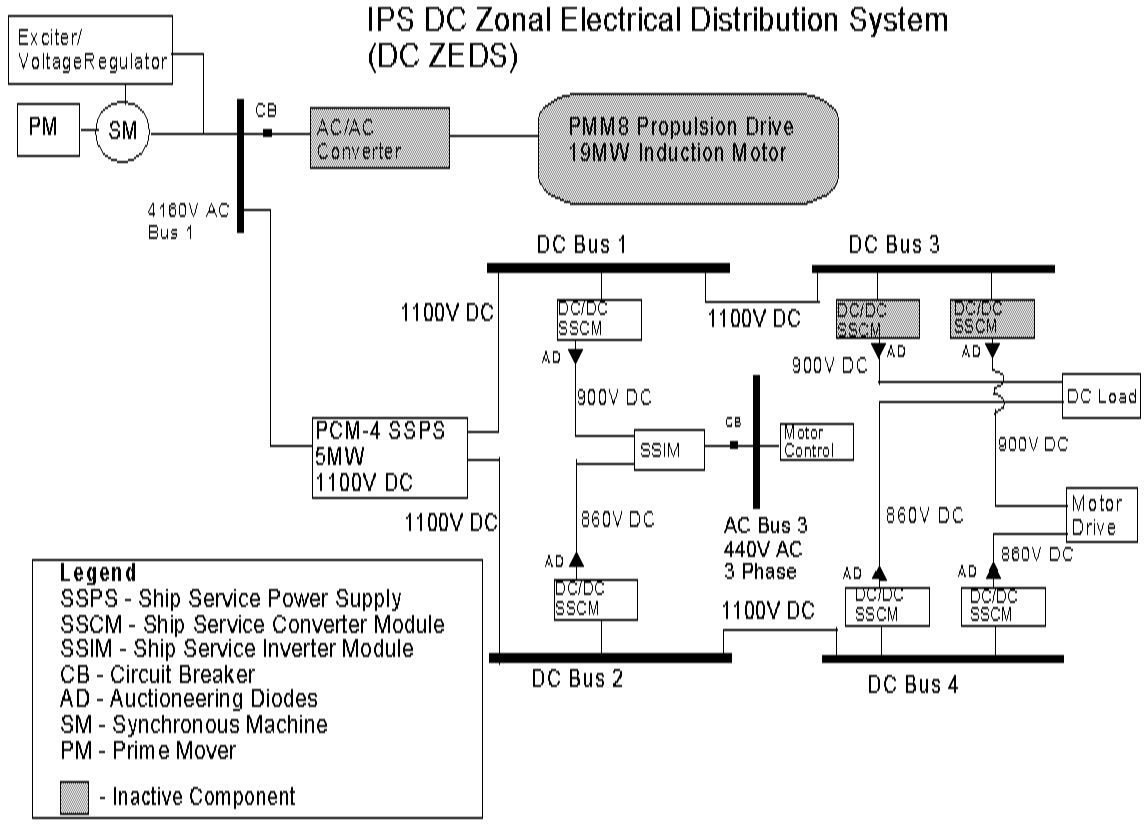


Figure 5-6 Run 4 Configuration

5.3.6 Run 5 Configuration

The configuration for Run 5 included energizing DC Buses 1, 2, 3, and 4. AC Bus 3 was also energized and connected to both DC Buses 1 and 2. However, DC Bus 3 was not connected to the motor.

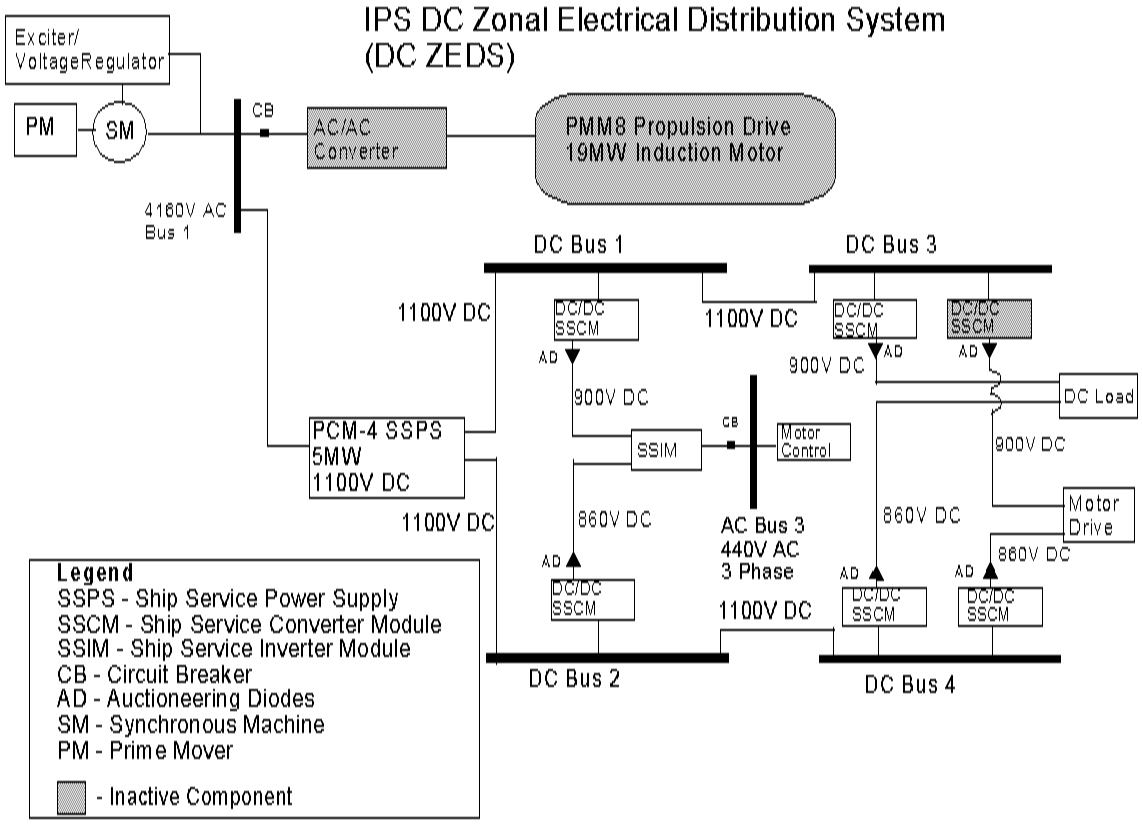


Figure 5-7 Run 5 Configuration

5.3.7 Run 6 Configuration

All of the DC Buses and AC Bus 3 were energized and connected. Run 6 is the complete power system operating.

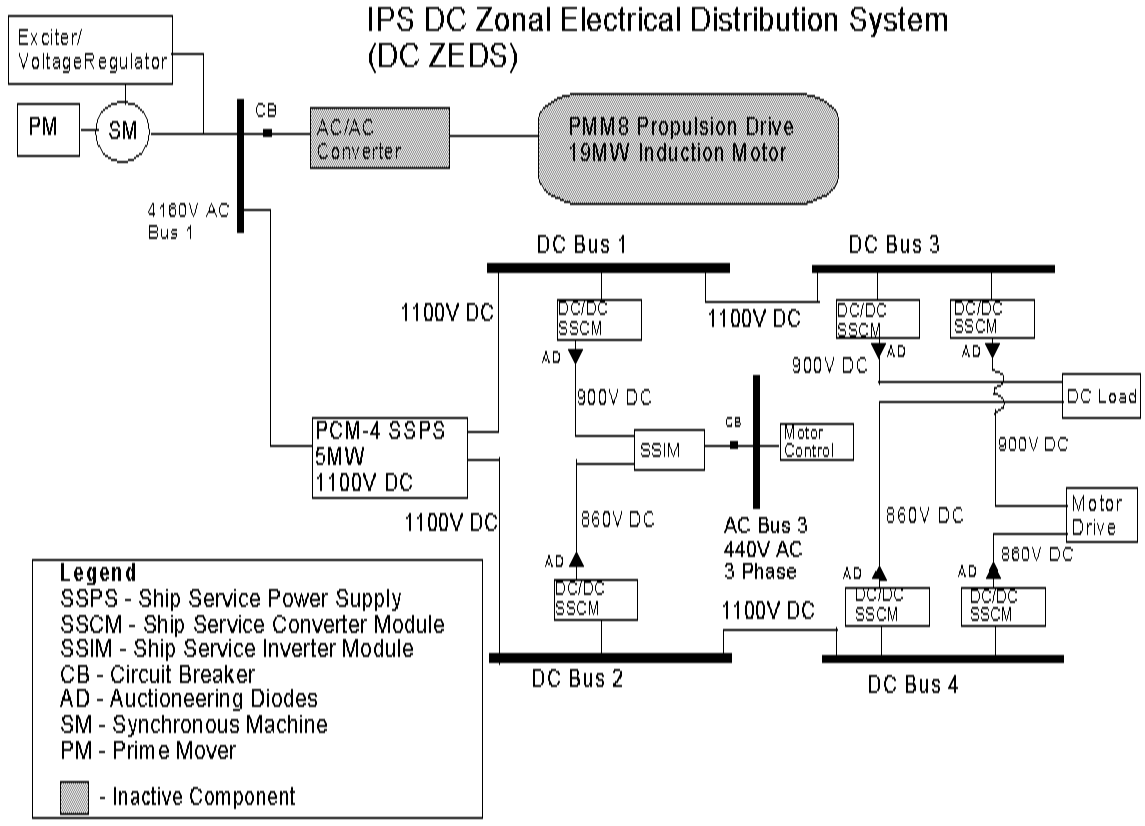


Figure 5-8 Run 6 Configuration

5.3.8 Run 7 Configuration

DC Bus 4 was disconnected from DC Bus 2 for Run 7. However, the rest of the system remained connected and energized.

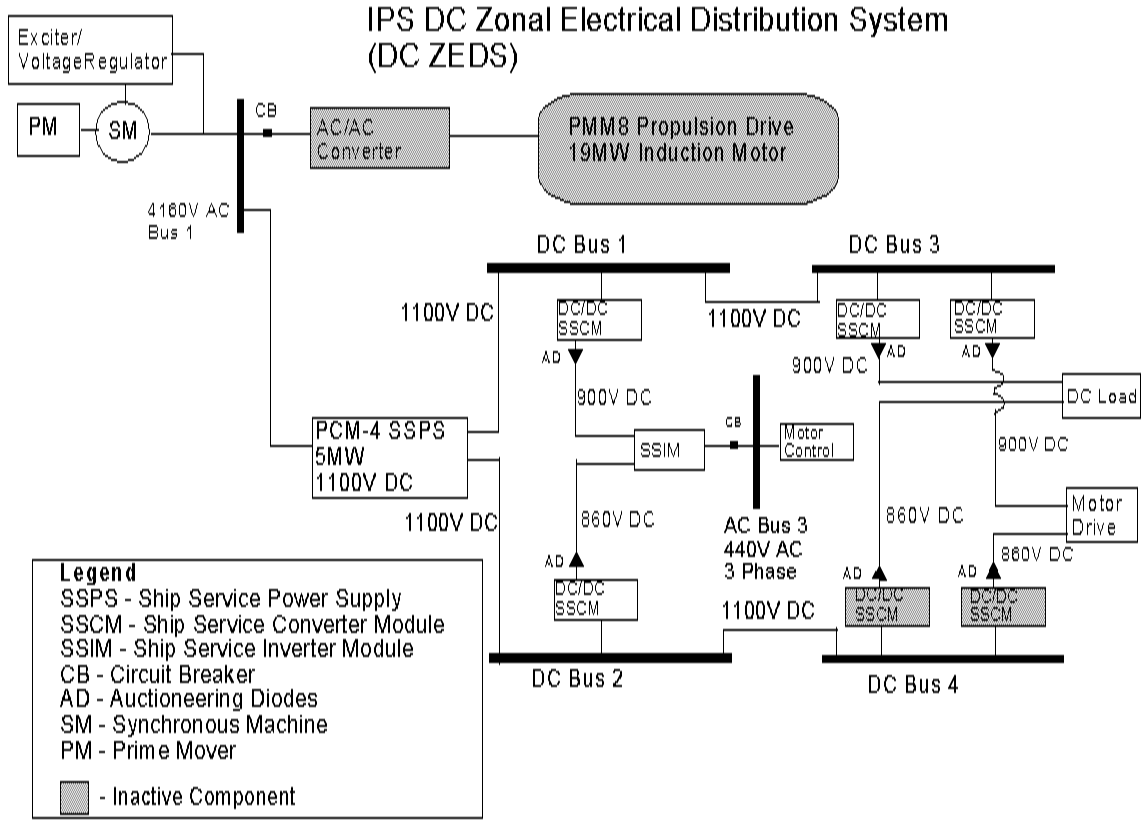


Figure 5-9 Run 7 Configuration

5.3.9 Run 8 Configuration

In this final run, DC Buses 2 and 4 were disconnected while DC Buses 1 and 3 remained energized. AC Bus 3 was still connected to DC Bus 1.

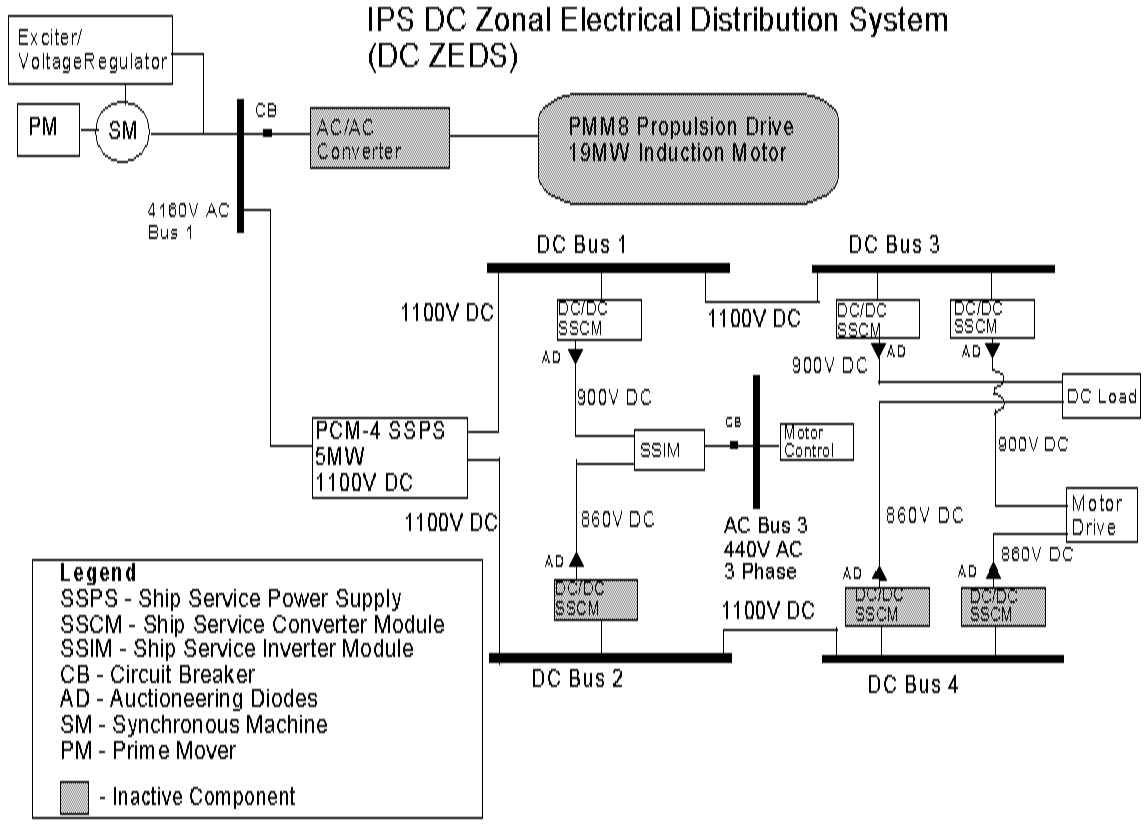


Figure 5-10 Run 8 Configuration

The IPS Simulation Code and the command file to exercise the simulation are found in Appendices B and C, respectively.

6 Variable Structure Control of IPS

It is necessary for the IPS controller to contend with a varying physical structure within the power system. An adept controller will have to adapt to such modifications in the power system with ease. During this investigation, three different neural network architectures were evaluated. At first, these networks were trained on millions of passes. Testing on independent data indicated the neural network's tendency to memorize the data instead of generalizing. When this memorization occurred, the performance of the neural network rapidly diminished. Consequently, the training of these variable structure networks was repeated using a reduced number of passes. This training strategy limited the memorization of the data by the neural networks and increased their robustness. The following sections present each neural network and provide graphical results. Quantitative and summary results are reported after discussion of individual neural network structures and results.

6.1 38-1 Variable Structure Neural Network

As shown in Figure 6-1, the 38-1 neural network has a 38 neuron input layer, a ten neuron hidden layer, and a single neuron output layer. The hidden layer uses a hyperbolic tangent transfer function. The network was trained using the Extended Delta-Bar-Delta (EDBD) learning rule. The 38 neuron input layer accepted data from 19 of the 20 parameters and their time-shifted equivalents. The auctioneering diode output, idc34b, is the current supplied to the motor during the start sequence. This network trained for 10,000 passes on runs 2, 3, 4, 6, 7, and 8. It was then tested on runs 1 (Figure 6-2) and 5 (Figure 6-3) without ever having trained on such configurations. Given the difficulty of modeling variable structure systems, the results exceed our expectations. Both runs very closely approximate the expected values, testifying to the robustness of the neural network.

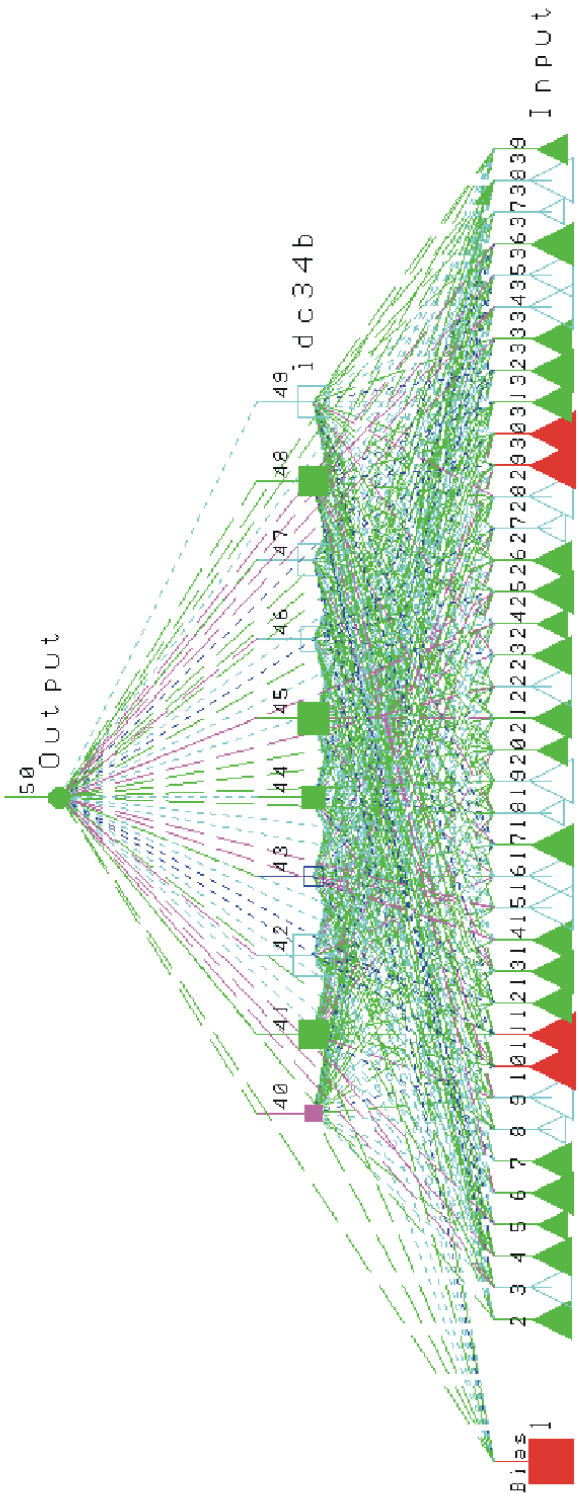


Figure 6-1 Neural Network with 38-1 Architecture

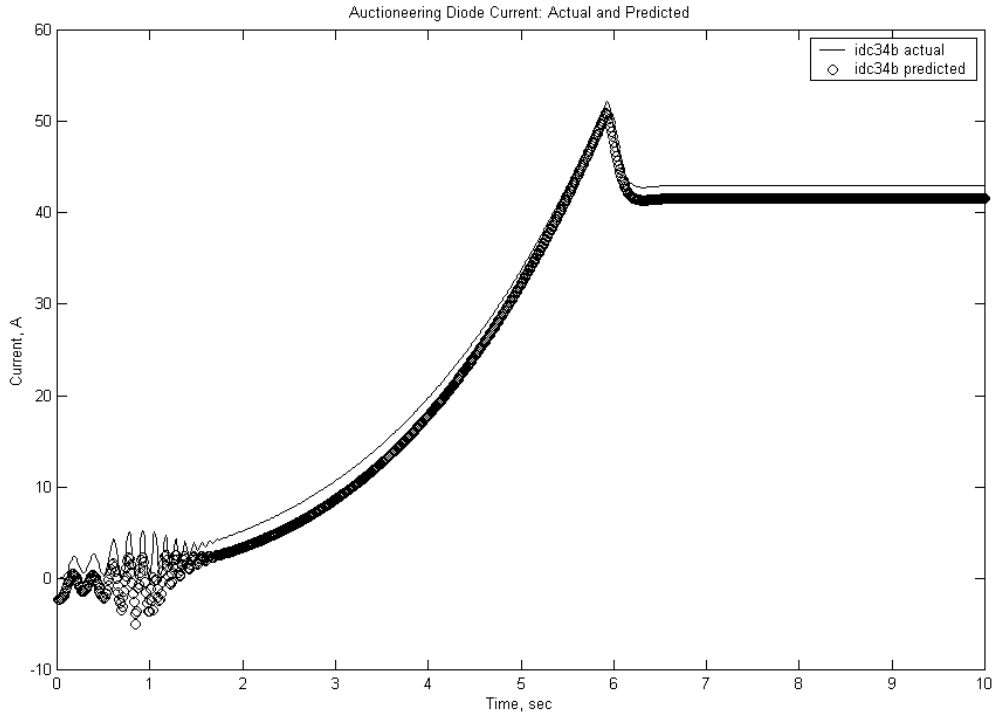


Figure 6-2 Run 1 Current Prediction with 38-1 Architecture

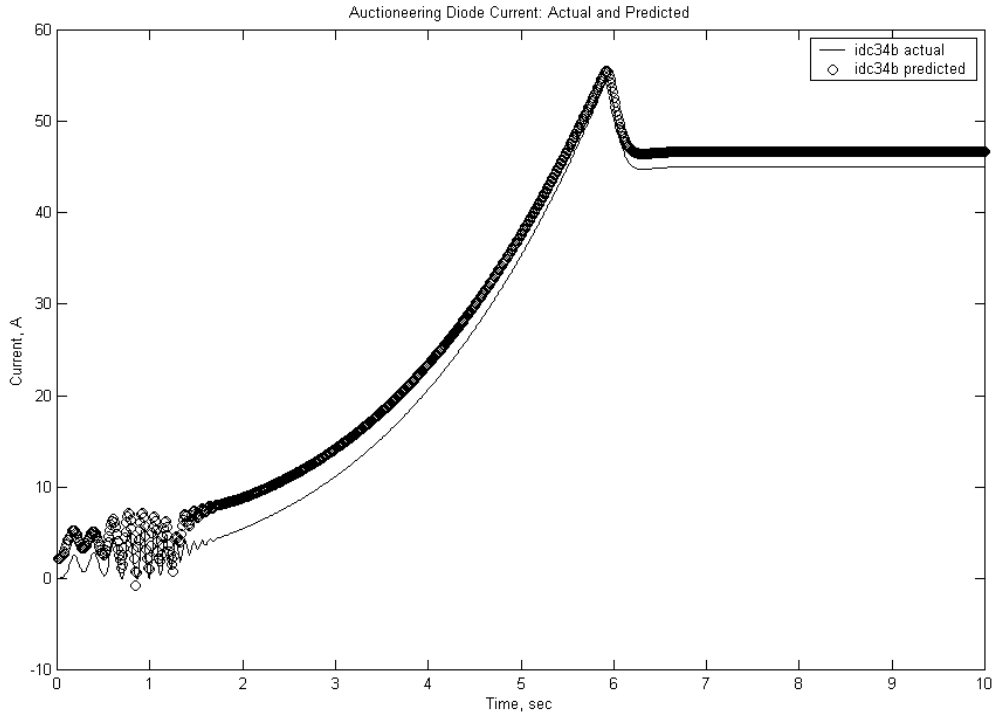


Figure 6-3 Run 5 Current Prediction with 38-1 Architecture

6.2 36-2 Variable Structure Neural Network

The 36-2 neural network, discussed in this section and illustrated in Figure 6-4, represents the next logical progression in predictive capability. This network uses all inputs except vdc34b and idc34b to predict both vdc34b and idc34b. As before, auctioneering diode output, idc34b and vdc34 represent the current and voltage supplied to the motor during the start sequence. The 36-2 nomenclature indicates 18 inputs plus 18 time shifted inputs and two outputs. The hyperbolic tangent transfer function was retained for hidden layers and once again, EDBD was the learning rule.

The 36-2 neural network was tested on runs 1 and 5 after having trained for 7,500 passes exclusively on runs 2, 3, 4, 6, 7, and 8. This network was not expected to provide accurate estimates for configurations that it had not seen during training. Nevertheless, Figures 6-5 and 6-6 indicate motor startup current predictions that are almost as good as the 38-1 case. The small offsets between actual and predicted results are due to the variable structure of the system. This observation is strengthened by examining the voltage predictions. Note that the predicted voltage for test scenarios 1 and 5 are identical, indicating that the neural network is unaware of the change in system configuration. The 36-2 network again demonstrated the robustness of neural networks in estimating idc34b and vdc34b effectively with minimal input information.

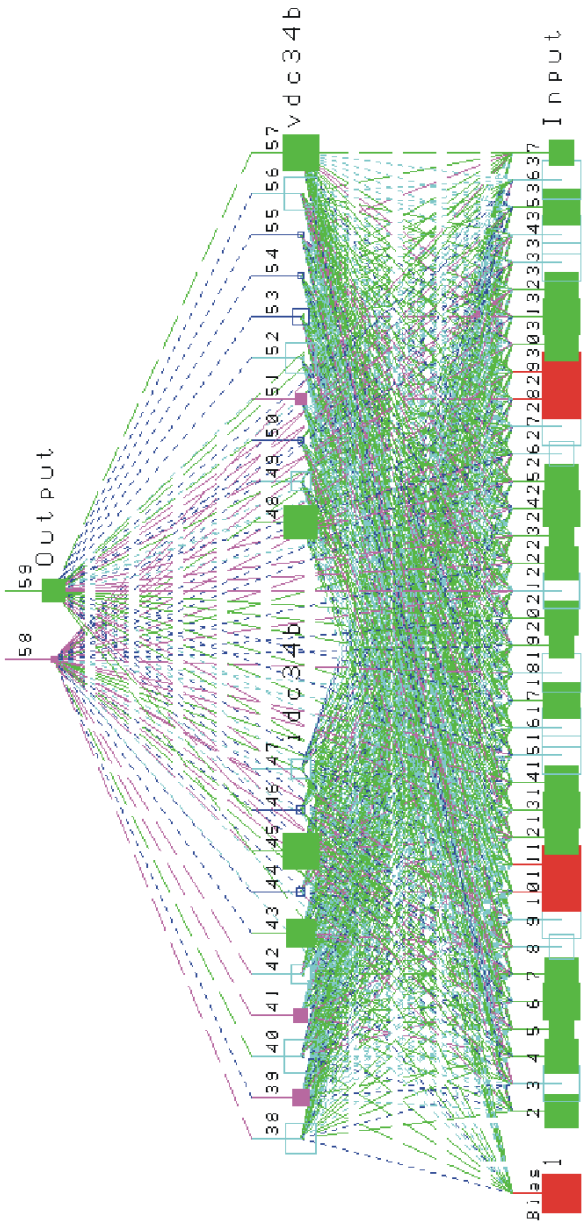


Figure 6-4 Neural Network with 36-2 Architecture

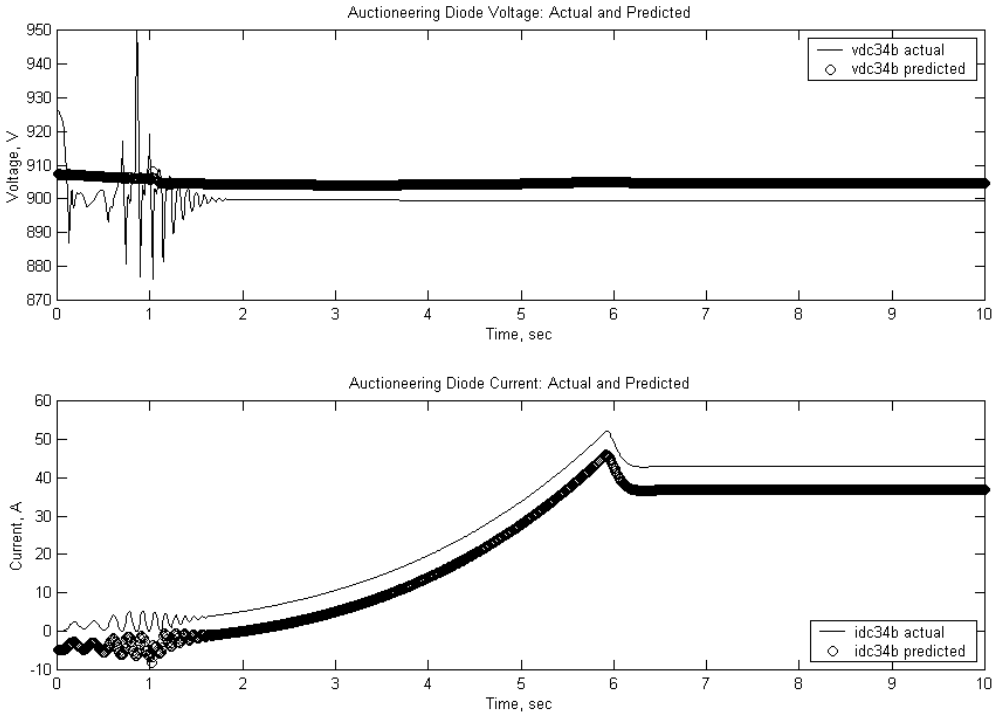


Figure 6-5 Run 1 Current Prediction with 36-2 Structure

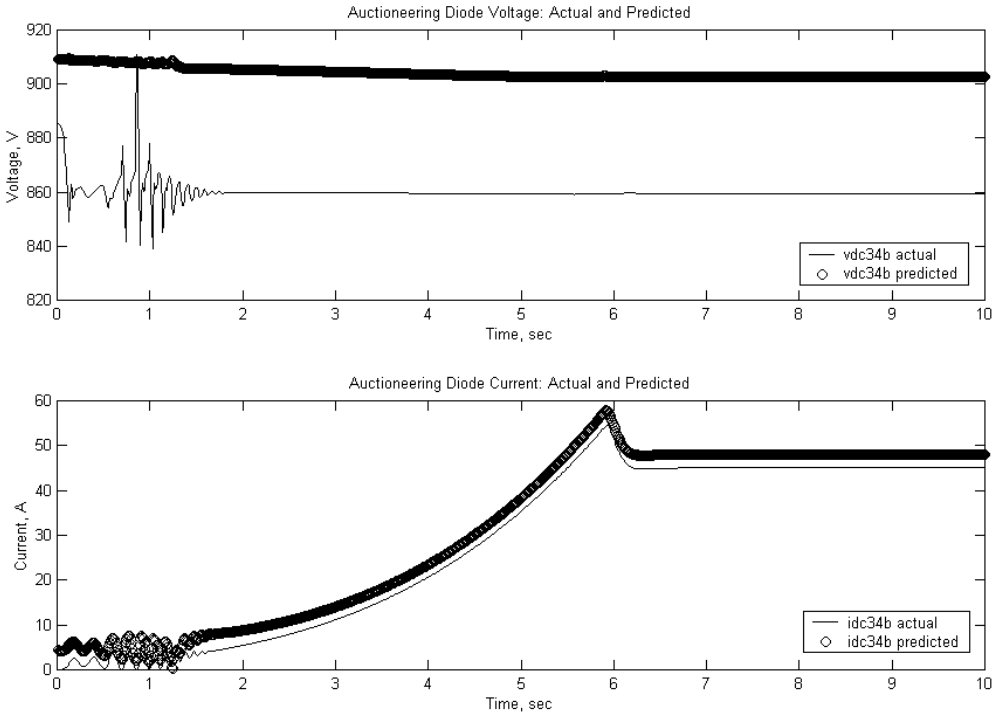


Figure 6-6 Run 5 Current Prediction with 36-2 Structure

6.3 40-20 Variable Structure Neural Network

To complete the variable structure studies, a 40-input 20-output (40-20) neural network was constructed. The 40-20 network is essentially twenty 38-1 networks acting in parallel. Since the 40-20 interconnections are very complex, Figure 6-7 displays a simplified schematic of the neural network. Each output estimates one of the twenty parameters that were observed in the simulations. To prevent memorization, each output is not connected to its own input or time-shifted input. Consequently, the neural network determines the output based upon the other 38 inputs. These 38 inputs are sent to a hidden layer that is directly connected to the corresponding single output neuron. In this network, the hidden layers all use the hyperbolic tangent transfer functions and apply the EDBD learning rule. This neural network is an attempt to extend the robustness of the 38-1 Variable Structure Neural Network into a controller that can determine all twenty system parameters.

The results for the 40-20 neural network are very good. However, as the neural network is trained on multiple output parameters, it tends to generalize. These generalizations limit the correlation of the 40-20 neural network as compared to the 38-1 neural network since all twenty output parameters in the 40-20 were weighted equally while the 38-1 network was solely concerned with idc34b.

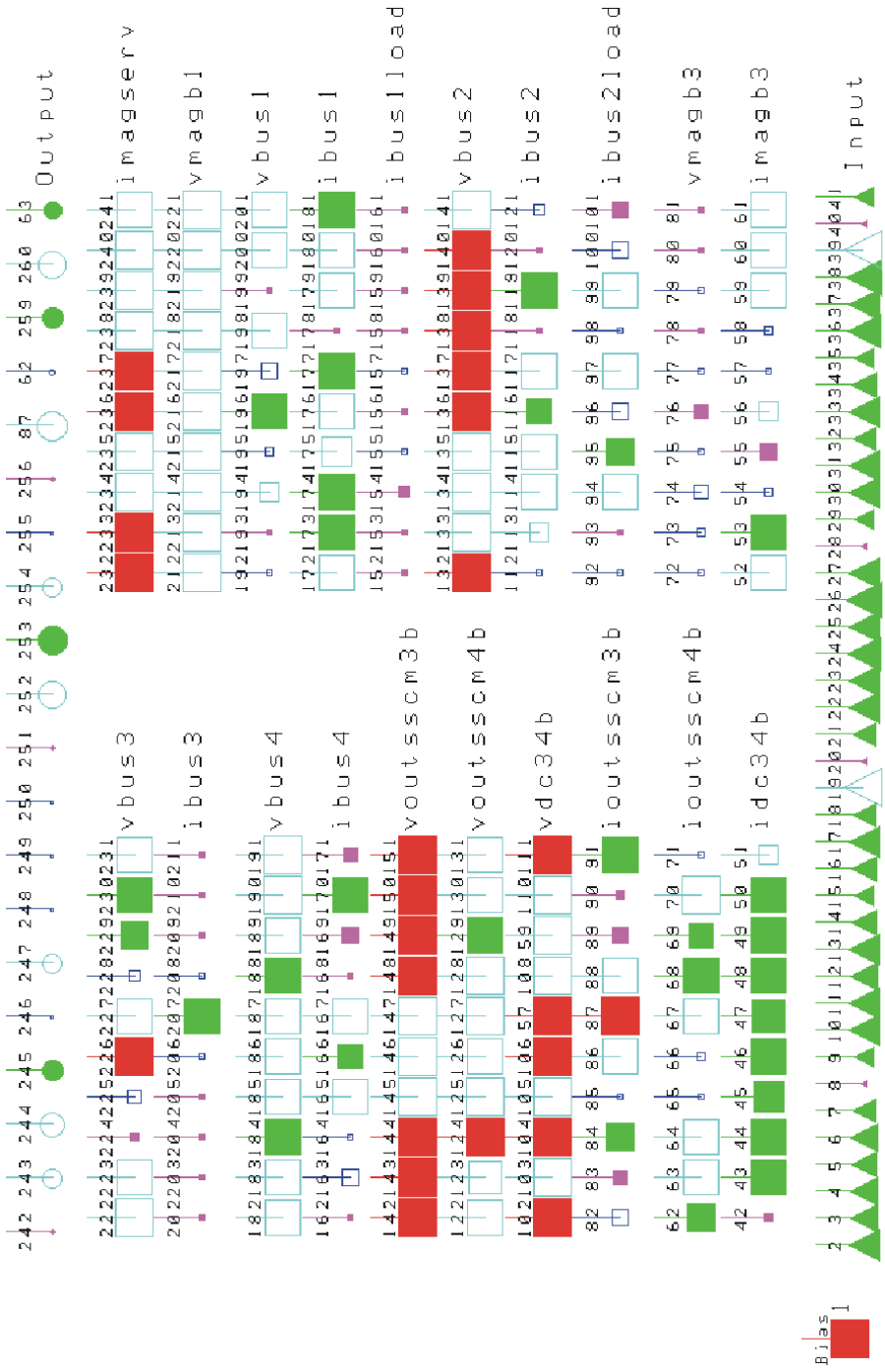


Figure 6-7 Neural Network: 40-20 Architecture

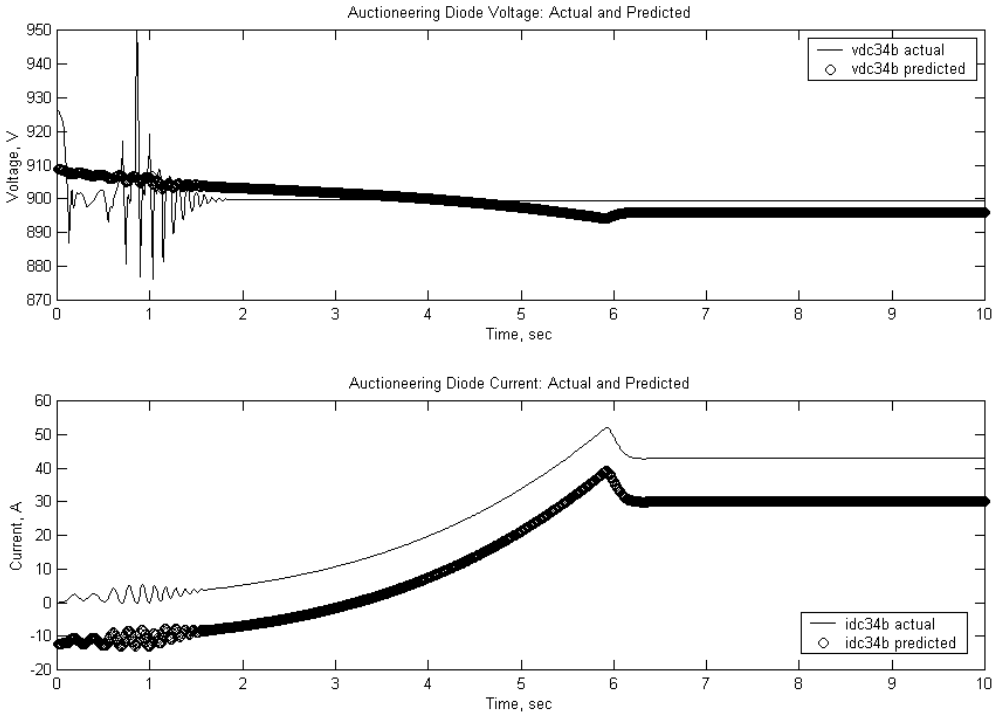


Figure 6-8 Run 1 Current & Voltage Prediction with 40-20 Architecture

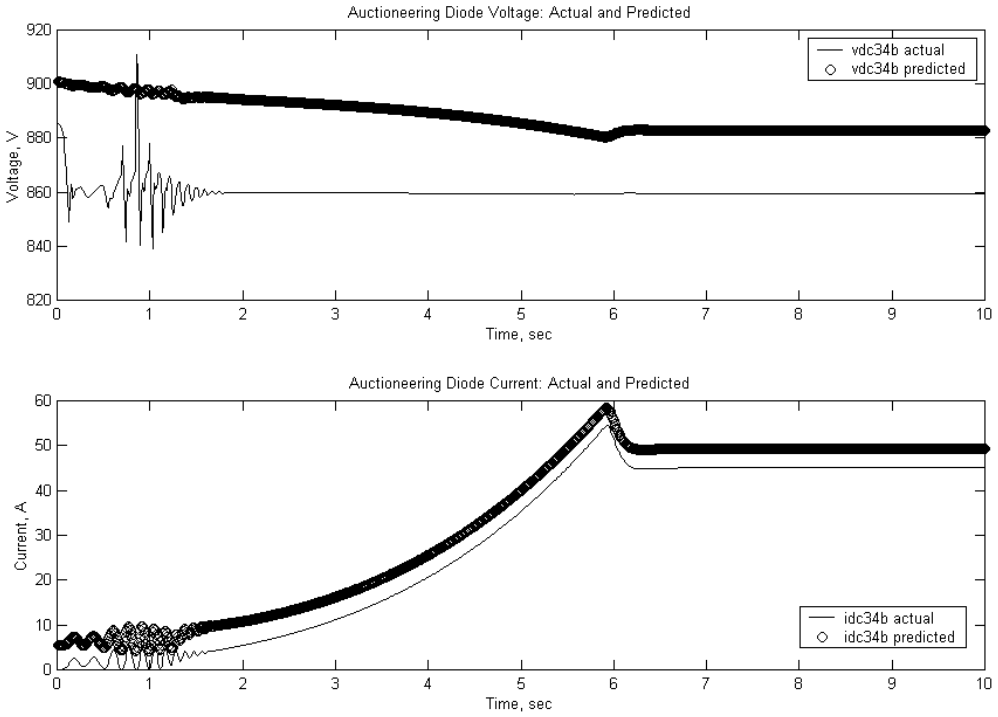


Figure 6-9 Run 5 Current & Voltage Prediction with 40-20 Architecture

7 Fault Tolerant Control of an IPS

A capable controller for the IPS must be fault tolerant. The controller cannot be so brittle that if sensor data degrades or is completely absent, it will fail. In order to address this concern, two fault tolerant neural networks were designed, developed, and tested.

7.1 38-1 Fault Tolerant Neural Network

The 38-1 neural network, diagrammed in Figure 7-1, has 38 input layer neurons that accept all of the parameters and their time-shifted equivalents except idc34b and its time-shifted equivalent. The variable idc34b was selected as the output since it is the current associated with the motor start sequence. The hidden layer contains ten neurons with hyperbolic tangent transfer functions.

The neural network was trained on perfect data from all eight runs for a total of 1,000 passes. It was then tested on a set of degraded data that originated from run 6. Run 6 data was processed by the MATLAB filter included in Appendix E which randomly inserted errors ranging from –10% to 10%. Then, columns of zeros were inserted into the file to simulate sensor failure. These zeros were inserted in both the parameter and its time shifted equivalent at the same time so that the data was consistent. The neural network was then tested on this set of degraded data. Since the original eight runs had established a very brittle neural network, it was unable to perform at an acceptable level during this test. Figure 7-2 displays the neural network's poor performance as was expected since the neural network had not yet been trained on inaccurate data. The weights obtained from the perfect data were used as initial weight values for data degradation training. The neural network was trained on degraded data from run 6 for 3,500 passes.

As shown in Figure 7-3, the neural network's performance was greatly enhanced and was able to handle both degraded data faults and sensor failure faults. There are two groups of data points clustered at 8 seconds and 10 seconds in Figure 7-3 that depart significantly from the actual values. These departures show the neural network's ability to partially compensate for sensor failure. With additional training on corrupted data, the neural network could learn to minimize the sensitivity to sensor failure thereby enhancing robustness.

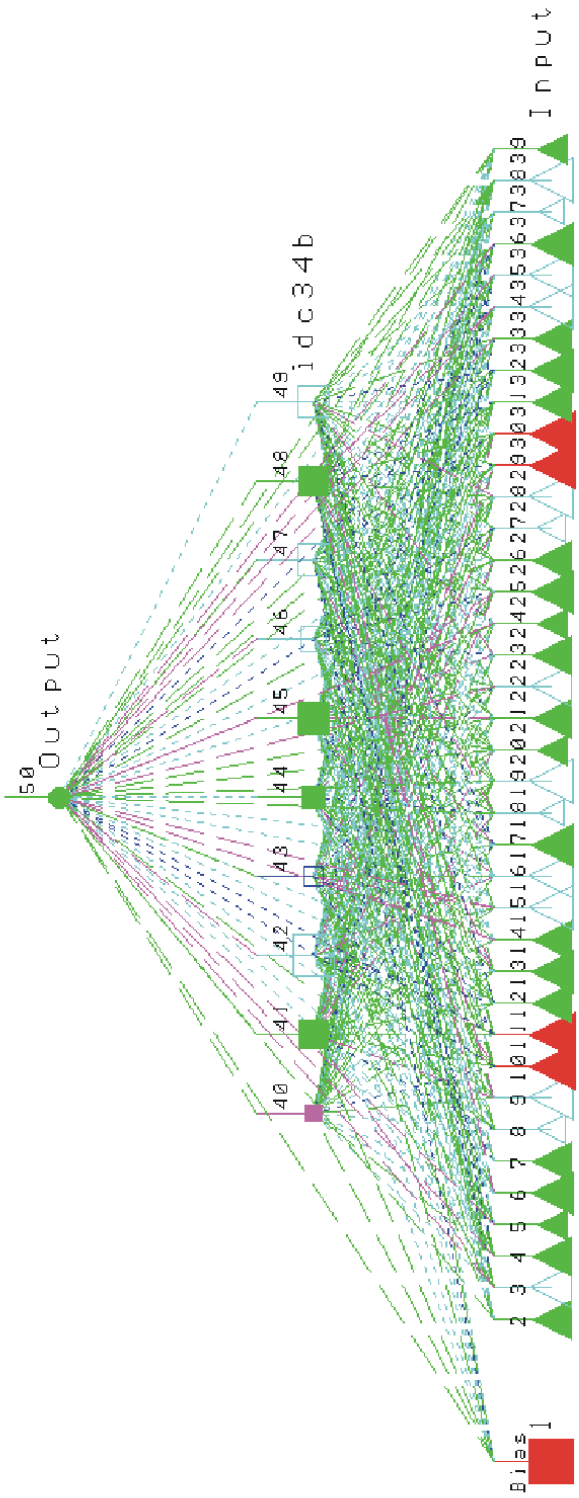


Figure 7-1 Neural Network with 38-1 Architecture

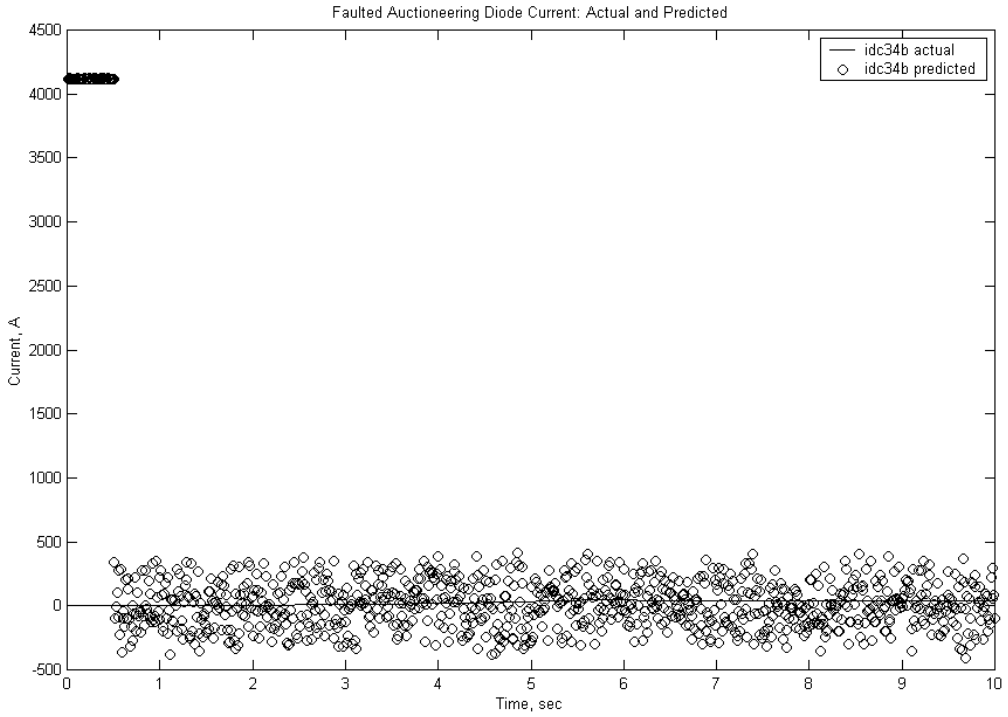


Figure 7-2 Initial Fault Tolerant Prediction with 38-1 Architecture

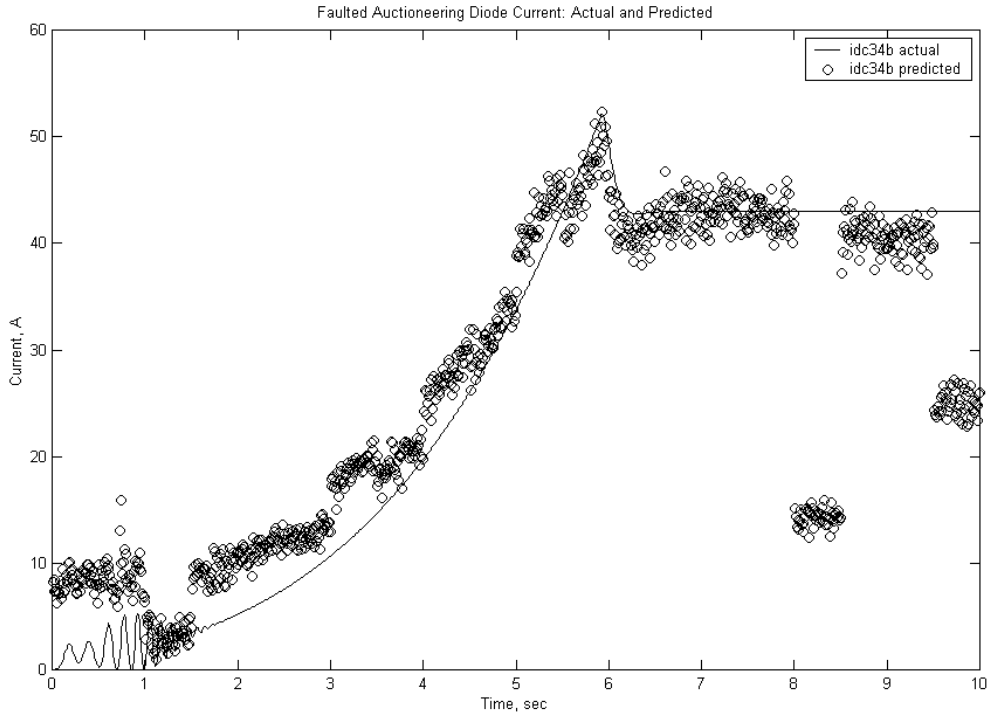


Figure 7-3 Improved Fault Tolerant Prediction with 38-1 Architecture

7.2 40-20 Fault Tolerant Neural Network

The 40 input and 20 output fault tolerant neural network is the culmination of this research. The primary goal of this project was to construct neural networks that could model nonlinear dynamics and be robust with respect to faulted data. This neural network is essentially twenty 38-1 neural networks combined to form one "super" neural network. Once again, the hidden layers are composed of hyperbolic tangent functions. Each output is a linear combination of the independent inputs and a dedicated hidden layer. Likewise, the hidden layer is connected to 38 input neurons excluding those that are directly related to the output. In order to show this neural network in the least confusing manner possible, the connections have been hidden in the Figure 7-4.

This network was trained in much the same way as the 38-1 Fault Tolerant Neural Network. It trained for 5 million passes on completely accurate data from all eight runs. As with the 38-1 neural network, a "naïve" 40-20 neural network that was not trained on faulted data could not predict faulted run 6 data. As before, training was continued using the degraded data developed for the 38-1 fault tolerant investigation. Due to the complexity of the 40-20 network, a total of 10 million passes was performed without over training. Figure 7-5 clearly demonstrates the ability of the 40-20 neural network to generate accurate estimates in spite of significant data degradation. The departures in Figure 7-5 are much smaller than those observed in Figure 7-3. The generalization of the 40-20 neural network decreased its sensitivity to sensor failure as compared to the 38-1 neural network in Figure 7-3. More detailed results are provided in Appendix H.

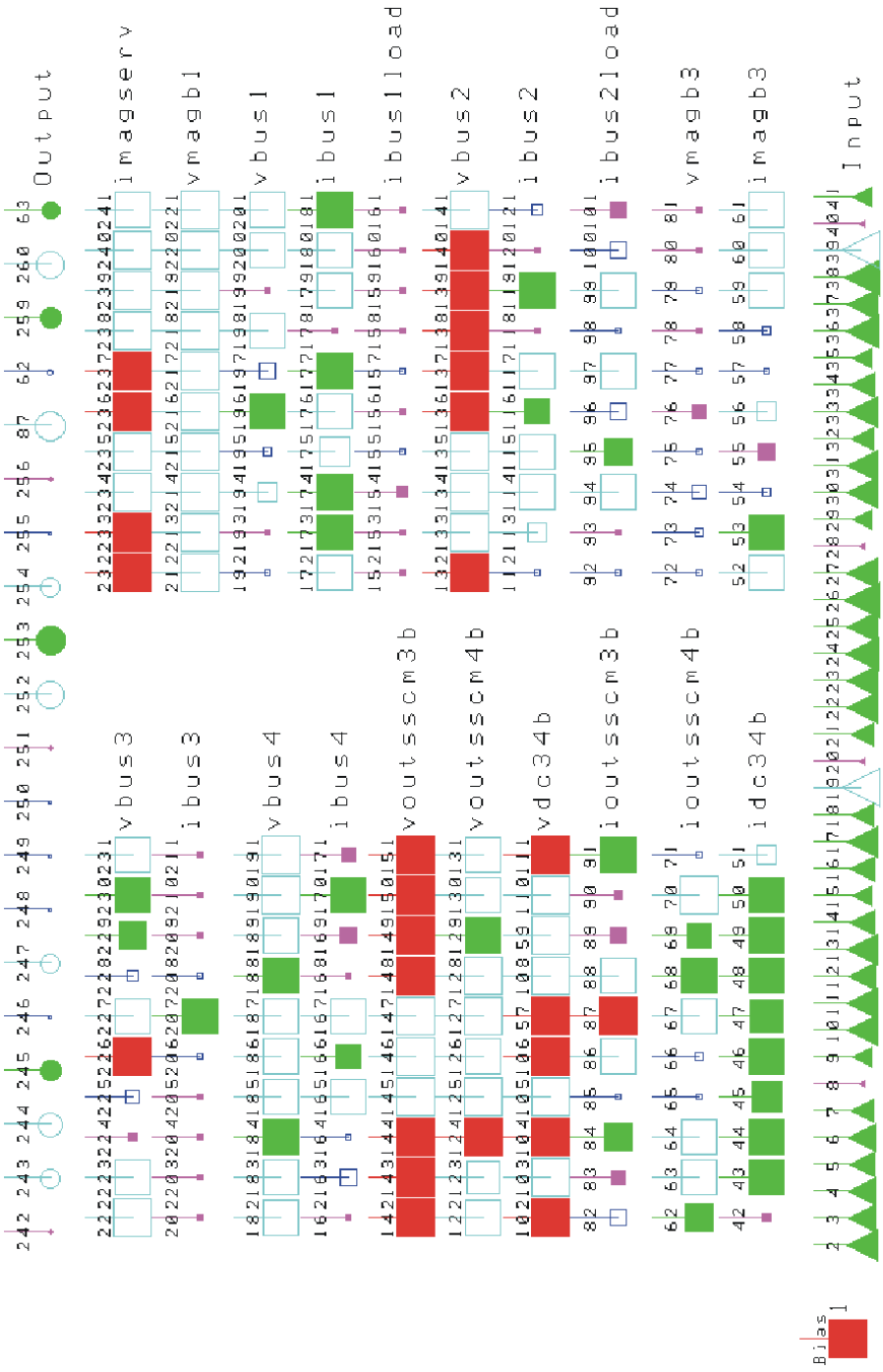


Figure 7-4 Neural Network with 40-20 Architecture

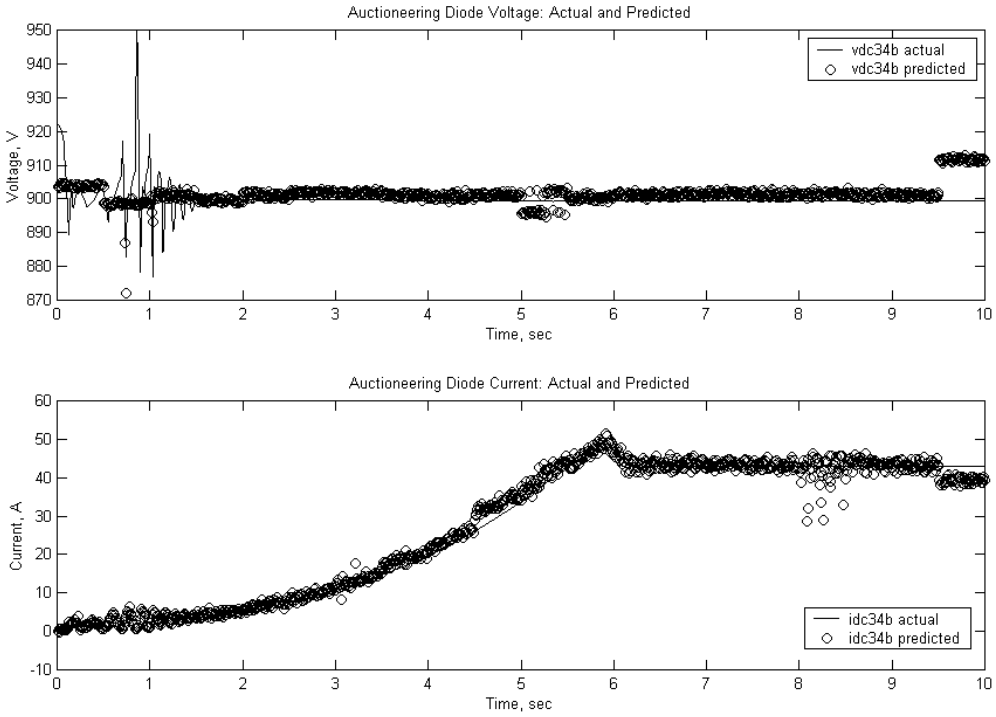


Figure 7-5 Fault Tolerant Prediction with 40-20 Architecture

8 Quantitative Analysis

It is just as important to measure the performance of the neural networks quantitatively (Table 8-1) as it is to demonstrate their performance graphically. Consequently, two types of calculations were applied to quantify the results: correlation and accuracy.

8.1 Correlation

Correlation measures the similarity of trends in two sets of data. If two sets of data are similar with respect to their slopes, they will have a correlation approaching 1. Their correlation will also be high even if there is a large bias separating the two sets of data. If two sets of data do not coincide with respect to their trends, then they will have a correlation approaching 0. Essentially, correlation measures the ability of the neural network to produce output data that tracks the trends of the desired output data without regard to scale. Therefore, correlation is not a useful metric for constant parameters.

8.2 Accuracy

For those sets of data that are constant or nearly constant, i.e. most voltage measurements in this research, an accuracy measurement was calculated using equation (8.1):

$$\text{accuracy} = 1 - \frac{\left|\bar{x}_{\text{desired}} - \bar{x}_{\text{predicted}}\right|}{x_{\text{nominal}}} \qquad (8.1)$$

This calculation included averaging the two sets of data, including the desired output and the predicted output, to be compared. Then, the absolute value of the difference between these two averages was taken and divided by a nominal value. This quantity was then subtracted from 1 to produce a measurement of accuracy. A value approaching 1 is very accurate while a value close to 0 is inaccurate. The nominal value for each parameter was selected from the design of the simulation. For bus voltages, the nominal value was the regulated voltage. For currents, a nominal value of 250 amperes was used.

8.3 Results

As one can deduce from these values in Table 8-1, the neural network was successful in predicting both transient and steady state responses. It should be noted that the 40-20 Variable Structure neural network failed to accurately predict vbus1 in run 1 and vbus2 and voutsscm3b in run 5. The neural network's poor performance with respect to these three parameters is a result of its lack of exposure to configurations in training that would have prepared it to predict these values. These results emphasize the importance of correctly training a neural network on multiple scenarios.

In transitioning from a simulation of the Integrated Power System to an operational Integrated Power System, the neural network's performance would be expected to improve. The greater amount of generalization necessary to observe an operational IPS would increase the neural network's robustness. An increase in sensors would also enhance the neural network's robustness. Finally, the computational expediency of the neural network would aid its performance in an operational Integrated Power System.

Table 8-1 Correlation & Accuracy Values

		Fault Tolerant				Variable Structure									
		40-20	40-20	38-1		40-20	40-20	40-20	40-20	36-2	36-2	36-2	36-2	38-1	38-1
		Run 6	Run 6	Run 6		Run 1	Run 1	Run 5	Run 5	Run 1	Run 1	Run 5	Run 5	Run 1	Run 5
Parameter	Nominal Value	Corr.	Acc.	Corr.		Corr.	Acc.	Corr.	Acc.	Corr.	Acc.	Corr.	Acc.	Corr.	Corr.
idc34b	250	0.9941		0.8756		0.9998		0.9998		0.9970		0.9996		0.9980	0.9950
vdc34b	900		0.9983				0.9912		0.9688		0.9947		0.9510		
imagserv	250	0.9332				0.9965		0.9964							
vmagb1	4160		0.9998				0.9975	0.9270							
vbus1	1200		0.9937				0.1007	0.9365							
ibus1	250	0.9890				0.9987		0.9201							
ibus1load	250		0.9999				0.9512		0.9834						
vbus2	1200	0.9178					0.7029		0.0578						
ibus2	250		0.9999				0.7276	0.9976							
ibus2load	250		0.9999				0.9330		0.9510						
vmagb3	440		0.9999				0.8581		0.8916						
imagb3	250		0.9999				0.8500		0.9903						
vbus3	1200		0.9995				0.9812		0.8485						
ibus3	250		0.9999				0.9522		0.8570						
vbus4	1200		0.9999				0.9756		0.8070						
ibus4	250		0.9999				0.9532		0.7628						
voutsscm3b	900		0.9967				0.8869		0.0318						
voutsscm4b	900		0.9964				0.9070		0.9440						
ioutsscm3b	250	0.9922				0.9998			0.9809						
ioutsscm4b	250		0.9999				0.9780	0.9999							

9 Summary

9.1 Synopsis

Neural network control of the Integrated Power System was researched, investigated, and evaluated from two distinct perspectives. The variable structure neural network was trained on six different power system configurations and evaluated using two additional, independent configurations. The training set was intentionally incomplete. This incompleteness was desired in order to fully assess the neural network's ability to analyze variable structure configurations. Despite the absence of prior training, the neural network accurately predicted the dynamics of the independent test configurations. The three isolated prediction failures are directly traceable to the absence of relevant training data. This result emphasizes the requirement for a complete training set. Subsequently, sensor degradation and failure were introduced in addition to the variable structure. The neural network continued to predict parameter values accurately.

9.2 Conclusions

The neural network did an excellent job of predicting uncertain power system parameters in a variable structure system subjected to degraded sensor data. The isolated estimation failures demonstrate that the training data must represent all possible system configurations. Given these very promising results, neural networks appear to have a great potential to aid in the fault tolerant control of complex, interactive systems such as the IPS.

9.3 Recommendations

In order to apply these promising results confidently to complex, interactive systems, additional research should be conducted in the following areas:

- Unexpected transients such as power system faults
- Quantification of fault tolerance
- Optimization of neural network structure, learning rules, and training
- Adding feedback structures to the neural network architecture
- Synthesize a hybrid control strategy where neural networks complement other control technologies

9.4 Closure

This research has produced exciting results that validate the premise that neural networks can be reliable, fault tolerant estimators of uncertain, complex, variable structure systems.

10 Endnotes

1. Liu, Chih-Wen, Mu-Chun Su, Shuenn-Shing Tsay, Yi-Jen Wang. "Application of a Novel Fuzzy Neural Network to Real-Time Transient Stability Swings Prediction Based on Synchronized Phasor Measurements" IEEE Transaction on Power Systems 14 (1999) pp 685-692.

2. Ba-Razzouk, Adellfattah, Ahmed Cheriti, Guy Olivie, Pierre Sicard. "Field Oriented Control of Induction Motors Using Neural-Network Decouplers" IEEE Transactions on Power Electronics 12 (1997) pp 752-763.

3. Guglielmi, G., T. Parisini, G. Rossi. "Fault Diagnosis and Neural Networks: A Power Plant Application" Control Engineering Practice 3 (1995) pp 601-620.

4. Barron, Roger L. and B. Eugene Parker, Jr. "Applicability of Neural Networks to Smart Shipboard Systems", Barron and Associates, 11 June 1993.

5. Zivi, Edwin L., Timothy J. McCoy. "Control of a Shipboard Integrated Power System" Procedings, 33rd Conference on Information Sciences and Systems, Johns Hopkins University, March 18, 1999.

6. Haykin, Simon. Neural Networks: A Comprehensive Foundation. Englewood Cliffs: Macmillan, 1994 p 2.

7. Narendra, K. S., K. Parthasarathy. "Identification and Control of Dynamical Systems Using Neural Networks" IEEE Transactions on Neural Networks 1 (1990) pp 4-27.

8. Neural Computing: A Technology Handbook for Professional II/Plus and NeuralWorks Explorer. Pittsburgh, NeuralWare, 1998.

9. NeuralWorks Advanced Reference Guide: A Software Reference for NeuralWorks Professional II/Plus and NeuralWorks Explorer, Pittsburgh, NeuralWare, 1998.

10. NeuralWorks Reference Guide: NeuralWorks Professional II/Plus and NeuralWorks Explorer. Pittsburgh, NeuralWare, 1998.

11. Barron, Roger L., Richard L. Cellucci, Paul R. Jordan, III, Norman E. Beam, Paul Hess and Andrew Barron. "Application of Polynomial Neural Networks to FDIE and Reconfigurable Flight Control." National Aerospace Electronics Conference, Dayton OH, 23-25 May1990.

12. Glover, S.F. , B.T., Kuhn., and S.D. Sudhoff, "IPS Reference Model", Purdue University, 16 April 1999.

13. AEgis Simulation, Inc. Advanced Continuous Simulation Language (ACSL): Reference Manual. Huntsville: Aegis Simulation, 1999.

11 Bibliography

Antsaklis, Panos J. "Neural Networks for Control Systems." IEEE Transactions on Neural Networks. 1 (1990) pp 242-244.

Barron, Roger L., Richard L. Cellucci, Paul R. Jordan, III, Norman E. Beam, Paul Hess and Andrew Barron. "Application of Polynomial Neural Networks to FDIE and Reconfigurable Flight Control." National Aerospace Electronics Conference, Dayton OH, 23-25 May1990.

Barron, Roger L. and B. Eugene Parker, Jr. "Applicability of Neural Networks to Smart Shipboard Systems.", Barron and Associates, 11 June 1993.

Barron and Associates, Inc. "Unified Open Architecture for Intelligent Ship Systems Automation." Vol. 1: Technical Proposal and Management Approach.

Ba-Razzouk, Adellfattah, Ahmed Cheriti, Guy Olivie, Pierre Sicard. "Field Oriented Control of Induction Motors Using Neural-Network Decouplers." IEEE Transactions on Power Electronics 12 (1997) pp 752-763.

Glover, S. F., B. T. Kuhn, and S. D. Sudhoff. "IPS System Description.", Purdue University, 16 April 1999.

Guglielmi, G., T. Parisini, G. Rossi. "Fault Diagnosis and Neural Networks: A Power Plant Application." Control Engineering Practice 3 (1995) pp 601-620.

Haykin, Simon. Neural Networks: A Comprehensive Foundation. Englewood Cliffs: Macmillan, 1994.

Kumar, D. M. Vinod, S. C. Srivastava, S. Shah, S. Mathur. "Topology Processing and Static State Estimation Uisng Aritficial Neural Networks." IEE Proceedings-Generation, Transmission and Distribution 143 (1996) pp 99-105.

Liu, Chih-Wen, Mu-Chun Su, Shuenn-Shing Tsay, Yi-Jen Wang. "Application of a Novel Fuzzy Neural Network to Real-Time Transient Stability Swings Prediction Based on Synchronized Phasor Measurements." IEEE Transaction on Power Systems 14 (1999) pp 685-692.

Lu, Songwu, Tamar Basar. "Robust Nonlinear System Identification Using Neural Network Models." IEEE Transactions on Neural Networks. 9 (1998) pp 407-429.

Lively, Kenneth, Tracey Thompson, Timothy J. McCoy, Edwin Zivi. "Adavnced Control Concepts for an Integrated Power System (IPS) Warship."

Narendra, K. S., K. Parthasarathy. "Identification and Control of Dynamical Systems Using Neural Networks." IEEE Transactions on Neural Networks 1 (1990) pp 4-27.

Neural Computing: A Technology Handbook for Professional II/Plus and NeuralWorks Explorer. Pittsburgh: NeuralWare, 1998.

NeuralWorks Advanced Reference Guide: A Software Reference for NeuralWorks Professional II/Plus and NeuralWorks Explorer. Pittsburgh: NeuralWare, 1998.

NeuralWorks Reference Guide: NeuralWorks Professional II/Plus and NeuralWorks Explorer. Pittsburgh: NeuralWare, 1998.

Nikiforov, I., V. Varavva, V. Kireichikov. "Application of Statistical Fault Detection Algorithms to Navigation System Monitoring." Automatica 3 (1993) pp 1275-1290.

Rao, Valluru, Hayagriva Rao. C++ Neural Networks and Fuzzy Logic, 2nd Edition. New York: MIS:Press, 1995.

Sudhoff, S. D., D.H. Schmuker, R. A. Youngs, H.J. Hegner. "Stability Analysis of DC Distribution Systems Using Admittance Space Constraints."

Sun, Y., H. Jiang, D. Wang. "Fault Synthetic Recognition for an EHV Transmission Line Using a Group of Neural Networks with a Time-Space Property." IEE Proceedings-Generation, Transmission and Distribution 145 (1998) pp 265-270.

Zivi, Edwin L. Submission of NARC Research Program "Dependable Automation Systems." 16 November 1998.

Zivi, Edwin L., Timothy J. McCoy. "Control of a Shipboard Integrated Power System" Procedings, 33rd Conference on Information Sciences and Systems, Johns Hopkins University, March 18, 1999.

12 Appendices

12.1 Appendix A – Preliminary Simulation MATLAB Code

```
% Preliminary Investigation - Simulation Data for Neural Network
% Single Matrix Output with Shifted Data [y1 y2]
% 27 MAR 00

clear;

%Define Lumped Parameters
ea = 10;
C1 = 1e-3;
L1 = 1;
R1 = 10;
R2 = 10;

%Derive the State-Space Matrix
A  = [-1/(R1*C1) -1/C1;1/L1 -R2/L1];
B  = [1/(C1*R1);0];
C  = [ea 0;0 ea;-ea/R1 -ea; ea -ea*R2;-ea/R1 0];
D  = [0;0;ea/R1;0;ea/R1];
sys = ss(A,B,C,D);
t=0:.001:.25;
U = ones(length(t),1);
XO=[0 0]';
[y,ts] = lsim(sys,U,t,XO);

% Single Matrix Output y
e1 = y(:,1);
i3 = y(:,2);
i2 = y(:,3);
e2 = y(:,4);

i1=  y(:,5);

% Shifted Matrix Output y_a
y1=y(1:length(y)-1,:);
y2=y(2:length(y),:);
y_a=[y1 y2];

%Plot simulation
clf
[ax1,h1,h2] = plotyy(t,i1,t,e1);
set(h1,'LineStyle','o')
set(h2,'LineStyle','-')
set(h2,'Marker','.')
set(h1,'Color','k')
set(h2,'Color','k')
title('Simulation Data for Neural Network')
xlabel('Time, sec.')
axes(ax1(1));
```

```
axis([0 max(t), -0.2 1])
ylabel('Current, A')
grid; hold on
plot(t,i2,'k+')
plot(t,i3,'kdiamond')
legend('i1','i2','i3',0)
axes(ax1(2));
axis([0 max(t), -2 10])
set(h2,'LineStyle','-.')
ylabel('Voltage, V')

grid; hold
plot(t,e2,'k:')
legend('e1','e2',4)

%Faulty Data
z=zeros(size(e1));
save sim1dat;
```

12.2 Appendix B – IPS ACSL Code

```
!-------------------------------------------------------------------!
!                                                                   !
! Authors:   S.F. Glover, B.T. Kuhn, S.D. Sudhoff                   !
!            Purdue University                                      !
!            Department of Electrical and Computer Engineering      !
!            1285 Electrical Engineering Building                   !
!            West Lafayette, IN 47907                               !
!            (765)-494-3246                                         !
! For:       Roger A. Dougal                                        !
!            University of South Carolina                           !
!            Department of Electrical and Computer Engineering      !
!            Columbia, South Carolina  29208                        !
!            Award # 97-364 - Modeling and Translator Development   !
! Date:      4/16/99                                                !
! Version:   2.0                                                    !
!                                                                   !
! Modified:  E.L. Zivi for 1/C Cerrito Trident Research 3/27/00     !
!                                                                   !
!-------------------------------------------------------------------!

INCLUDE 'macros/asscm.mac'
INCLUDE 'macros/diobravm.mac'
INCLUDE 'macros/mc.mac'
INCLUDE 'macros/dcresld.mac'
INCLUDE 'macros/inductqd.mac'
INCLUDE 'macros/rotor.mac'
INCLUDE 'macros/speedld.mac'
INCLUDE 'macros/dctxline.mac'
INCLUDE 'macros/assim.mac'
INCLUDE 'macros/apwsp.mac'
INCLUDE 'macros/revtrans.mac'
INCLUDE 'macros/srf.mac'
INCLUDE 'macros/aexciter.mac'
INCLUDE 'macros/asmsat.mac'
INCLUDE 'macros/aacbus.mac'
INCLUDE 'macros/pmm8.mac'

PROGRAM scsystem

   DYNAMIC

      ALGORITHM IALG=2
      MAXTERVAL MAXT=1.0e-4
      MINTERVAL MINT=1.0e-20
      CINTERVAL CINT = 1.0e-3
      CONSTANT TSTOP = 1.0
    VARIABLE T,TIC=0.0
      TERMT(t .GE. tstop-0.5*cint,'Exit on Tstop')

      DERIVATIVE main
```

```
!----------AC GENERATION--------------------------------------!

   !Turbine
    CONSTANT wrsm=377.0
    CONSTANT we=377.0
    SRF(1,we,qe)

   !Synchronous Machine IEEE Type 2 Excitation System"
    AEXCITER(ex,vqdsesm, vfd, &
                "vrefex=4160.0","vfdicex=22.1354", &
                "trex=1.05e-3","kaex=380.0","taex=0.019", &
                "vrminex=0.0","vrmaxex=7.3", &
                "kfex=31.5e-3","tf1ex=0.95","tf2ex=0.96", &
                "keex=0.95","teex=0.84", &
                "se100ex=0.903","se75ex=0.525", &
                "vbsmex=2.52e3","vfdbsmex=28.77")

   !Salient Synchronous Machine With 1 Damper"
    ASMSAT(sm,vqde1,we,vfd,wrsm,cbsm,vqdsesm, &
              iqdesm,ifda1,tesm, &
              "vfdicsm=22.1354","delta0sm=0.0", &
              "rssm=1.399e-3","llssm=372.115e-6", &
              "lmqsm=2.769e-3", &
              "llkqsm=141.66e-6","rkqsm=5.80e-3", &
              "llkdsm=72.05e-6", "rkdsm=5.08e-3", &
              "llfdsm=236.392e-6", "rfdsm=470.34e-6", &
              "nsfdsm=6.328e-2","npsm=2.0", &
              "masm=21899.65","mdsm=21585.83", &
              "tautsm=7.396","lamtsm=11.01","ithreshsm=1.0e7")


!----------AC Bus 1-------------------------------------------!

   ! Bus Voltage Magnitude
     CONSTANT root3o2=1.2247
     vmagb1 = SQRT(vqde1(1)**2 + vqde1(2)**2)*root3o2

     PROCEDURAL(imagb1, imagprop, imagserv = iqdesm, iqimprop, &
                                             idimprop, iqdepwsp)
   ! Bus 1 Generator Supply Current Magnitude
     imagb1 = SQRT(iqdesm(1)**2 + iqdesm(2)**2)*root3o2
   ! Bus 1 Propulsion Load Current Magnitude
     imagprop = SQRT(iqimprop**2 + idimprop**2)*root3o2
   ! Bus 1 Ships Service Load Current Magnitude
     imagserv = SQRT(iqdepwsp(1)**2 + iqdepwsp(2)**2)*root3o2
     END !procedural

   ! Calculate the bus voltage, - rbus is about 10 pu
     AACBUS(bs, iqdebc1, vqde1, &
              "rbs=2.622","tau1bs=1.0e-6","tau2bs=1.0e-4")
     PROCEDURAL(iqdebc1 = iqdesm,iqdepwsp,iqdemcprop)
       SUM(iqdebc1 = iqdesm,iqdepwsp,iqdemcprop)
     END !procedural
```

```
!---------PROPULSION SYSTEM-----------------------------------!

   ! Propulsion controller
     PMM8(mcprop,opmcprop,vqde1,we,iqimprop,idimprop, &
                  wrmimprop,wrmstarprop, &
                  vqimprop,vdimprop,weimprop,iqdemcprop, &
                  "cdcmcprop=0.2","taumcprop=0.1", &
                  "rsmcprop=8.83e-3","Lssmcprop=10.073e-3", &
                  "wbemcprop=30.02","vlinkicmcprop=0.0", &
                  "vlinkminmcprop=4250.0", &
                  "accllmcprop=-0.1241","acculmcprop=0.1241", &
                  "npmcprop=4.0","kpmcprop=1.0", &
                  "kimcprop=1.0","vsmaxmcprop=3396.6", &
                  "ramptmcprop=10.0", &
                  "lacmcprop=6.786e-6","rlcmcprop=0.0", &
                  "vscrmcprop=0.0","ldcmcprop=3.0e-3", &
                  "rdcmcprop=6.786e-3")

   ! 19MW induction motor
     INDUCTQD(improp, vqimprop,vdimprop,weimprop,wrmimprop, &
                     iqimprop,idimprop,teimprop, &
                     "rsimprop=8.83e-3","Llsimprop=173.1e-6", &
                     "Lmimprop=9.9e-3","rrpimprop=63.5e-3", &
                     "Llrpimprop=173.1e-6","nPimprop=4.0", &
                     "lamdsicimprop=0.0","lamdrpicimprop=0.0")
     ROTOR(improp, teimprop,tlimprop, wrmimprop, &
                   "Jimprop=3.698e6","wrmicimprop=0.0")
     SPEEDLD(improp, wrmimprop, tlimprop, &
                    "wrmbaserpmimprop=148.1","tlbaseimprop=1.23e6")


!---------DC POWER SUPPLY-------------------------------------!

   CONSTANT vrefpwsp= 1100.0
   CONSTANT odflagpwsp = .TRUE.
   CONSTANT cnntpwsp1 = .TRUE.
   CONSTANT cnntpwsp2 = .TRUE.
   INITIAL
      cbpwsp = .TRUE.
      oppwsp =. FALSE.
   END !initial
   APWSP(pwsp, oppwsp,cbpwsp,odflagpwsp,cnntpwsp1, &
         cnntpwsp2,vrefpwsp,vqde1, &
         vbus1,vbus2,we,qe, itranpwsp1,itranpwsp2,iqdepwsp, &
         "rppwsp=52.06e-3","llppwsp=2.07e-3", &
         "rspwsp=.069","llspwsp=7.61e-3","npwsp=9.8", &
         "vscrpwsp=0.0","ldcpwsp=128.0e-6", &
         "rdcpwsp=0.19e-3","cdcpwsp=17.6e-3", &
         "kpvpwsp=0.011","kivpwsp=22.0","limvpwsp=10000.0", &
         "ilimitpwsp=10000.0","kpipwsp=0.012","kiipwsp=0.0", &
         "limipwsp=0.0","alphaminpwsp=0.0", &
         "alphamaxpwsp=3.14","vmaxpwsp=1175.0", &
         "taufltpwsp=4.364e-3","dismodepwsp=.FALSE.", &
         "ithreshpwsp=1.0e7","L1txpwsp=10.0e-6", &
         "r1txpwsp=0.01","L2txpwsp=10.0e-6","r2txpwsp=0.01")
```

```
!---------DC BUS 1-----------------------------------!

   !calculate bus voltage
   pvbus1 = (itranpwsp1 - itran13 - ilinsscm1a)/Cinsscm1a
   vbus1 = INTEG(pvbus1, 0.0)

   PROCEDURAL(ibus1, ibusload = itranpwsp1, ioutsscm1a)
ibus1 = itranpwsp1
ibus1load = ioutsscm1a
END ! Procedural


   !sscm1a
   ASSCM(sscm1a, opsscm1a,vbus1,ioutsscm1a, &
                 ilinsscm1a,Cinsscm1a,voutsscm1a, &
                 "vinstarsscm1a=1100.0","vrefsscm1a=900.0", &
                 "Cin1sscm1a=500.0e-6","Cin2sscm1a=100.0e-6", &
                 "Coutsscm1a=1.0e-3","Linsscm1a=70.0e-6", &
                 "Loutsscm1a=100.0e-6","kpsscm1a=2.0e-3", &
                 "kisscm1a=0.7","fswsscm1a=20.0e3", &
                 "tausscm1a=7.96e-6","vswsscm1a=1.0", &
                 "vdiodesscm1a=1.0","RLinsscm1a=0.01", &
                 "RLoutsscm1a=0.02","RCoutsscm1a=1.0", &
                 "trampsscm1a=30.0e-3")


!---------DC BUS 2-----------------------------------!

   !calculate bus voltage
   pvbus2 = (itranpwsp2 - itran24 - ilinsscm2a)/Cinsscm2a
   vbus2 = INTEG(pvbus2, 0.0)

   PROCEDURAL(ibus2, ibus2load = itranpwsp2, ioutsscm2a)
ibus2 = itranpwsp2
ibus2load = ioutsscm2a
END ! Procedural


   !sscm2a
   ASSCM(sscm2a, opsscm2a,vbus2,ioutsscm2a,  &
                 ilinsscm2a,Cinsscm2a,voutsscm2a, &
                 "vinstarsscm2a=1100.0","vrefsscm2a=860.0", &
                 "Cin1sscm2a=500.0e-6","Cin2sscm2a=100.0e-6", &
                 "Coutsscm2a=1.0e-3","Linsscm2a=70.0e-6", &
                 "Loutsscm2a=100.0e-6","kpsscm2a=2.0e-3", &
                 "kisscm2a=0.7","fswsscm2a=20.0e3", &
                 "tausscm2a=7.96e-6","vswsscm2a=1.0", &
                 "vdiodesscm2a=1.0","RLinsscm2a=0.01", &
                 "RLoutsscm2a=0.02","RCoutsscm2a=1.0", &
                 "trampsscm2a=30.0e-3")


!--------------LOADS FED FROM BUSSES 1 & 2----------!
```

```
      !~~~~~~position a~~~~~~!
         !diode bridge
         DIOBRAVM(db12a, voutsscm1a,voutsscm2a,vdc12a,faultdb1a,faultdb2a,
&
                         ioutsscm1a,ioutsscm2a,idc12a, &
                         "L1db12a=1.0e-6","r1db12a=0.01", &
                         "L2db12a=1.0e-6","r2db12a=0.01")

         !SSIM
         ASSIM(ssim12a, opssim12a,iqim12a,idim12a,idc12a, &
                        voqssim12a,vodssim12a,vdc12a, &
                        "cdcssim12a=1000.0e-6", &
                        "rlfssim12a=0.05","lfssim12a=20.0e-6", &
                        "cfssim12a=30.0e-6","rfssim12a=0.2", &
                        "voutstarssim12a=359.26","wessim12a=377.0", &
                        "cfestssim12a=30.0e-6", &
                        "kpvssim12a=0.8","kivssim12a=25.0", &
                        "kpissim12a=0.11","kiissim12a=750.0", &
                        "ilimitssim12a=742.07")

         !50 HP induction motor
         INDUCTQD(im12a, voqssim12a,vodssim12a,wessim12a,wrmim12a, &
                         iqim12a,idim12a,teim12a, &
                         "rsim12a=0.087","Llsim12a=0.0008011", &
                         "Lmim12a=0.03469","rrpim12a=0.228", &
                         "Llrpim12a=0.0008011","nPim12a=4.0", &
                         "lamdsicim12a=0.0","lamdrpicim12a=0.0")
         ROTOR(im12a, teim12a,tlim12a, wrmim12a, &
                       "Jim12a=1.662","wrmicim12a=0.0")
         SPEEDLD(im12a, wrmim12a, tlim12a, &
                        "wrmbaserpmim12a=1705.0","tlbaseim12a=198.0")

   !---------AC BUS 3-----------------------------------!

    PROCEDURAL (vmagb3, imagb3 = voqssim12a, vodssim12a, iqim12a, idim12a)
      ! Bus 3 Voltage Magnitude
        vmagb3 = SQRT(voqssim12a**2 + vodssim12a**2)*root3o2
      ! Bus 3 Load Current Magnitude
        imagb3 = SQRT(iqim12a**2 + idim12a**2)*root3o2
      END !procedural

   !-------Transmission Lines--------------------------!

      !transmission line between busses 1 and 3
      DCTXLINE(tran13, optran13,vbus1,vbus3, itran13, &
                       "Ltran13=10.0e-6","Rtran13=0.01")

      !transmission line between busses 2 and 4
      DCTXLINE(tran24, optran24,vbus2,vbus4, itran24, &
                       "Ltran24=10.0e-6","Rtran24=0.01")


   !---------DC BUS 3-----------------------------------!
```

```
   !calculate bus voltage
   pvbus3 = (itran13 - ilinsscm3a - ilinsscm3b)/(Cinsscm3a + Cinsscm3b)
   vbus3 = INTEG(pvbus3, 0.0)

   PROCEDURAL(ibus3 = ioutsscm3a)
ibus3  = ioutsscm3a
END ! Procedural

   !sscm3a
   ASSCM(sscm3a, opsscm3a,vbus3,ioutsscm3a, &
                 ilinsscm3a,Cinsscm3a,voutsscm3a, &
                 "vinstarsscm3a=1100.0","vrefsscm3a=900.0", &
                 "Cin1sscm3a=500.0e-6","Cin2sscm3a=100.0e-6", &
                 "Coutsscm3a=1.0e-3","Linsscm3a=70.0e-6", &
                 "Loutsscm3a=100.0e-6","kpsscm3a=2.0e-3", &
                 "kisscm3a=0.7","fswsscm3a=20.0e3", &
                 "tausscm3a=7.96e-6","vswsscm3a=1.0", &
                 "vdiodesscm3a=1.0","RLinsscm3a=0.01", &
                 "RLoutsscm3a=0.02","RCoutsscm3a=1.0", &
                 "trampsscm3a=30.0e-3")

   !sscm3b
   ASSCM(sscm3b, opsscm3b,vbus3,ioutsscm3b, &
                 ilinsscm3b,Cinsscm3b,voutsscm3b, &
                 "vinstarsscm3b=1100.0","vrefsscm3b=900.0", &
                 "Cin1sscm3b=500.0e-6","Cin2sscm3b=100.0e-6", &
                 "Coutsscm3b=1.0e-3","Linsscm3b=70.0e-6", &
                 "Loutsscm3b=100.0e-6","kpsscm3b=2.0e-3", &
                 "kisscm3b=0.7","fswsscm3b=20.0e3", &
                 "tausscm3b=7.96e-6","vswsscm3b=1.0", &
                 "vdiodesscm3b=1.0","RLinsscm3b=0.01", &
                 "RLoutsscm3b=0.02","RCoutsscm3b=1.0", &
                 "trampsscm3b=30.0e-3")


!---------DC BUS 4-----------------------------------!

   !calculate bus voltage
   pvbus4 = (itran24 - ilinsscm4a - ilinsscm4b)/(Cinsscm4a + Cinsscm4b)
   vbus4 = INTEG(pvbus4, 0.0)

   PROCEDURAL(ibus4 = ioutsscm4a)
ibus4  = ioutsscm4a
END ! Procedural

   !sscm4a
   ASSCM(sscm4a, opsscm4a,vbus4,ioutsscm4a, &
                 ilinsscm4a,Cinsscm4a,voutsscm4a, &
                 "vinstarsscm4a=1100.0","vrefsscm4a=860.0", &
                 "Cin1sscm4a=500.0e-6","Cin2sscm4a=100.0e-6", &
                 "Coutsscm4a=1.0e-3","Linsscm4a=70.0e-6", &
                 "Loutsscm4a=100.0e-6","kpsscm4a=2.0e-3", &
                 "kisscm4a=0.7","fswsscm4a=20.0e3", &
                 "tausscm4a=7.96e-6","vswsscm4a=1.0", &
                 "vdiodesscm4a=1.0","RLinsscm4a=0.01", &
```

```
                    "RLoutsscm4a=0.02","RCoutsscm4a=1.0", &
                    "trampsscm4a=30.0e-3")

      !sscm4b
      ASSCM(sscm4b, opsscm4b,vbus4,ioutsscm4b, &
                    ilinsscm4b,Cinsscm4b,voutsscm4b, &
                    "vinstarsscm4b=1100.0","vrefsscm4b=860.0", &
                    "Cin1sscm4b=500.0e-6","Cin2sscm4b=100.0e-6", &
                    "Coutsscm4b=1.0e-3","Linsscm4b=70.0e-6", &
                    "Loutsscm4b=100.0e-6","kpsscm4b=2.0e-3", &
                    "kisscm4b=0.7","fswsscm4b=20.0e3", &
                    "tausscm4b=7.96e-6","vswsscm4b=1.0", &
                    "vdiodesscm4b=1.0","RLinsscm4b=0.01", &
                    "RLoutsscm4b=0.02","RCoutsscm4b=1.0", &
                    "trampsscm4b=30.0e-3")


   !--------------LOADS FED FROM BUSSES 3 & 4----------!

      !~~~~~~position a~~~~~~!
         !diode bridge
         DIOBRAVM(db34a, voutsscm3a,voutsscm4a,vdc34a,faultdb3a,faultdb4a,
&
                         ioutsscm3a,ioutsscm4a,idc34a, &
                         "L1db34a=1.0e-6","r1db34a=0.01", &
                         "L2db34a=1.0e-6","r2db34a=0.01")

         !resistive load
         DCRESLD(res34a, opres34a,idc34a, vdc34a, &
                         "cdcres34a=1000.0e-6","rdcres34a=4.05")

      !~~~~~~position b~~~~~~!
         !diode bridge
         DIOBRAVM(db34b, voutsscm3b,voutsscm4b,vdc34b,faultdb3b,faultdb4b,
&
                         ioutsscm3b,ioutsscm4b,idc34b, &
                         "L1db34b=1.0e-6","r1db34b=0.01", &
                         "L2db34b=1.0e-6","r2db34b=0.01")

         !induction motor controller
         MC(mc34b, opmc34b,idc34b,iqim34b,idim34b, &
                   wrmim34b,wrmstarim34b, &
                   vdc34b,vqim34b,vdim34b,weim34b, &
                   "cdcmc34b=100.0e-6","taumc34b=0.1", &
                   "rsmc34b=0.087","Lssmc34b=3.549e-2", &
                   "wbemc34b=377.0","vdcicmc34b=0.0", &
                   "accllmc34b=-4.0","acculmc34b=30.0", &
                   "npmc34b=4.0","kpmc34b=1.0", &
                   "kimc34b=20.0","vsmaxmc34b=375.5")

         !50 HP induction motor
         INDUCTQD(im34b, vqim34b,vdim34b,weim34b,wrmim34b, &
                         iqim34b,idim34b,teim34b, &
                         "rsim34b=0.087","Llsim34b=0.8011e-3", &
                         "Lmim34b=34.69e-3","rrpim34b=0.228", &
```

```
                        "Llrpim34b=0.8011e-3","nPim34b=4.0", &
                        "lamdsicim34b=0.0","lamdrpicim34b=0.0")
          ROTOR(im34b, teim34b,tlim34b, wrmim34b, &
                       "Jim34b=1.662","wrmicim34b=0.0")
          SPEEDLD(im34b, wrmim34b, tlim34b, &
                        "wrmbaserpmim34b=1705.0","tlbaseim34b=198.0")


      END !Derivative

      SRFD(1,qe)
      apwspd(pwsp,we,oppwsp)

      DISCRETE cbcontrol
        ASMSATD(sm,cbsm)
      END ! DISCRETE

    END ! Dynamic

END ! Program
```

12.3 Appendix C – IPS ACSL Simulation Commands

```
!-------------------------------------------------------------------!
!                                                                   !
! Authors:   S.F. Glover, B.T. Kuhn, S.D. Sudhoff                   !
!            Purdue University                                      !
!            Department of Electrical and Computer Engineering      !
!            1285 Electrical Engineering Building                   !
!            West Lafayette, IN 47907                               !
!            (765)-494-3246                                         !
! For:       Roger A. Dougal                                        !
!            University of South Carolina                           !
!            Department of Electrical and Computer Engineering      !
!            Columbia, South Carolina  29208                        !
!            Award # 97-364 - Modeling and Translator Development   !
! Date:      4/16/99                                                !
! Version:   1.0                                                    !
!                                                                   !
! Modified:  E.L. Zivi for 1/C Cerrito Trident Research 4/1/00      !
!                                                                   !
!-------------------------------------------------------------------!
s hvdprn=.f.
s weditg=.f. !write events to log file
s strplt=.t. !strip plots
s calplt=.f. !plots on one graph
s alcplt=.f. !no colored plots
s cjvitg=.f. !turns off checking of jacobian validity

output t,cioitg,cssitg /nciout=10

prepare t         ! needed for output command
prepare wrmimprop ! prop shaft speed

   !----------AC Bus 1-----------------------------------------------!
prepare imagb1    ! AC Bus 1 Generator Supply Current Magnitude
prepare imagprop  ! AC Bus 1 Propulsion Load Current Magnitude
! << neural net data starts here >>
prepare imagserv  ! AC Bus 1 Ships Service Load Current Magnitude
prepare vmagb1 ! AC Bus 1 Voltage Magnitude

   !---------DC BUS 1-----------------------------------------------!
prepare vbus1  ! DC Bus 1 voltage
prepare ibus1  ! DC Bus 1 total current (load + bus3)
prepare ibus1load ! DC Bus 1 load current

   !---------DC BUS 2-----------------------------------------------!
prepare vbus2  ! DC Bus 2 voltage
prepare ibus2  ! DC Bus 2 total current (load + bus3)
prepare ibus2load ! DC Bus 2 load current

   !---------AC BUS 3-----------------------------------------------!
prepare vmagb3 ! AC Bus 3 Voltage Magnitude
prepare imagb3 ! AC Bus 3 Load Current Magnitude
```

```
   !---------DC BUS 3-------------------------------------------!
prepare vbus3  ! DC Bus 3 voltage
prepare ibus3  ! DC Bus 3 total current = load current

   !---------DC BUS 4-------------------------------------------!
prepare vbus4  ! DC Bus 4 voltage
prepare ibus4  ! DC Bus 4 total current = load current

   !---------DC BUS 3/4 Load B Converters-------------------------!
prepare voutsscm3b ! voltage from Bus 3 to diode bridge
prepare voutsscm4b ! voltage from Bus 3 to diode bridge
prepare vdc34b  ! voltage out of auctioneering diodes
prepare ioutsscm3b ! current from Bus 3 to diode bridge
prepare ioutsscm4b ! current from Bus 4 to diode bridge
prepare idc34b  ! current out of auctioneering diodes
! << neural net data ends here >>

   !---------DC BUS 3/4 Load B Motor Drive-----------------------!
prepare wrmim34b     ! commanded motor speed, rad/sec
prepare wrmstarim34b ! actual motor speed, rad/sec

PROCEDURE initrun    ! establish standard initial alignment
   s tic=-5   ! need time to reach steady state
   action /var=tic/val=.t./loc=cnntpwsp1       ! connect DC bus 1
   action /var=tic/val=.t./loc=cnntpwsp2       ! connect DC bus 2
   action /var=tic/val=.t./loc=oppwsp      ! turn on AC/DC power supply
   action /var=tic/val=.t./loc=optran13    ! connect DC bus 1 -> 3
   action /var=tic/val=.t./loc=optran24    ! connect DC bus 2 -> 4
   action /var=tic/val=.f./loc=opsscm1a    ! ship service converters off
   action /var=tic/val=.f./loc=opsscm2a
   action /var=tic/val=.f./loc=opsscm3a
   action /var=tic/val=.f./loc=opsscm3b
   action /var=tic/val=.f./loc=opsscm4a
   action /var=tic/val=.f./loc=opsscm4b
   action /var=tic/val=.f./loc=opssim12a       ! ship service inverter off
   action /var=tic/val=.f./loc=opmcprop    ! propulsion off
   action /var=tic/val=180/loc=wrmstarim34b! cmd induction motor 1719 rpm
   action /var=tic/val=0.0/loc=wrmstarprop ! cmd prop. zero shaft speed
END !initrun

PROCEDURE plotrun ! standard plots for each run
   ! AC Bus 1 voltage, AC Bus 1, ship service, propulsion current
   plot/xlo=0.0 vmagb1/lo=4100/hi=4220,imagb1/lo=0/hi=7500, &
        imagserv/lo=0/hi=100,wrmimprop/lo=0/hi=15
   ! DC Bus 1,2 voltage, current
   plot/xlo=0.0 vbus1/lo=1000/hi=1200,ibus1/lo=0/hi=300, &
        vbus2/lo=1000/hi=1200,ibus2/lo=0/hi=300
   ! DC Bus 1,2 loads, AC Bus 3 voltage, current
   plot/xlo=0.0 ibus1load/lo=0/hi=50,ibus2load/lo=0/hi=50 &
        vmagb3/lo=400/hi=500,imagb3/lo=0/hi=100
   ! DC Bus 3,4 voltage, current
   plot/xlo=0.0 vbus3/lo=1000/hi=1200,ibus3/lo=0/hi=300, &
        vbus4/lo=1000/hi=1200,ibus4/lo=0/hi=300
   ! motor drive supply voltages, drive voltages, command
   plot/xlo=0.0 voutsscm3b/lo=600/hi=1000,voutsscm4b/lo=600/hi=1000, &
```

```
        vdc34b/lo=600/hi=1000,wrmstarim34b/lo=0/hi=200
   ! motor drive supply currents, drive current, motor speed
   plot/xlo=0.0 ioutsscm3b/lo=0/hi=100,ioutsscm4b/lo=0/hi=100, &
        idc34b/lo=0/hi=100,wrmim34b/lo=0/hi=200
END ! plotrun

PROCEDURE plotinit ! standard plots for each run w/ initial transients
   ! AC Bus 1 voltage, AC Bus 1, ship service, propulsion current
   plot/xaxis=t/xlo=-5 vmagb1/lo=0/hi=4300,imagb1/lo=0/hi=7500, &
                       imagserv/lo=0/hi=100,wrmimprop/lo=0/hi=15
   ! DC Bus 1,2 voltage, current
   plot/xaxis=t/xlo=-5 vbus1/lo=0/hi=1200,ibus1/lo=0/hi=300, &
                       vbus2/lo=0/hi=1200,ibus2/lo=0/hi=300
   ! DC Bus 1,2 loads, AC Bus 3 voltage, current
   plot/xaxis=t/xlo=-5 ibus1load/lo=0/hi=50,ibus2load/lo=0/hi=50 &
                       vmagb3/lo=0/hi=500,imagb3/lo=0/hi=100
   ! DC Bus 3,4 voltage, current
   plot/xaxis=t/xlo=-5 vbus3/lo=0/hi=1200,ibus3/lo=0/hi=300, &
        vbus4/lo=0/hi=1200,ibus4/lo=0/hi=300
   ! motor drive supply voltages, drive voltages, command
   plot/xaxis=t/xlo=-5 voutsscm3b/lo=0/hi=1000,voutsscm4b/lo=0/hi=1000, &
        vdc34b/lo=0/hi=1000,wrmstarim34b/lo=0/hi=200
   ! motor drive supply currents, drive current, motor speed
   plot/xaxis=t/xlo=-5 ioutsscm3b/lo=0/hi=100,ioutsscm4b/lo=0/hi=100, &
        idc34b/lo=0/hi=100,wrmim34b/lo=0/hi=200
END ! plotinit

PROCEDURE run1 ! IPS Motor Start: DC Bus 1,3 DC Load
   initrun    ! set generic initial conditions
   action /var=tic/val=.f./loc=cnntpwsp2       ! disconnect DC bus 2
   !turn on the system components, start motor at t = 0.0
   action /var=-4.9/val=.t./loc=opsscm3a       ! DC bus 3 converters on
   action /var=-4.9/val=.t./loc=opsscm3b
   action /var=-4.9/val=.t./loc=opres34a       ! DC bus 3/4 resistive load
   action /var=0.0/val=.t./loc=opmc34b     ! DC bus 3/4 induction motor

   !start the study
   s tstop=0.0, cint=1.0e-2, maxt(1)=5.0e-4
   start
   s tstop=10.0, cint=1.0e-2, maxt(1)=5.0e-4
   cont
   s title=120*" " ! clear title
   ! plot title 12345678901234567890123456789012345678901234567890
   s title(01)="IPS Motor Start Run 1"
   s title(41)="DC Bus 1,3 Energized"
   s title(81)="DC Load On"
   ! plotrun      ! standard plots
   ! print/noheader/file="run1.txt"/all
   action /clear
END !run1

PROCEDURE run2 ! IPS Motor Start: DC Bus 2,4 DC Load
   initrun    ! set generic initial conditions
   action /var=tic/val=.f./loc=cnntpwsp1       ! disconnect DC bus 1
   !turn on the system components, start motor at t = 0.0
```

```
   action /var=-4.9/val=.t./loc=opsscm4a       ! DC bus 4 converters on
   action /var=-4.9/val=.t./loc=opsscm4b
   action /var=-4.9/val=.t./loc=opres34a       ! DC bus 3/4 resistive load
   action /var=0.0/val=.t./loc=opmc34b     ! DC bus 3/4 induction motor

   !start the study
   s tstop=0.0, cint=1.0e-2, maxt(1)=5.0e-4
   start
   s tstop=10.0, cint=1.0e-2, maxt(1)=5.0e-4
   cont
   s title=120*" " ! clear title
   ! plot title 12345678901234567890123456789012345678901234567890
   s title(01)="IPS Motor Start Run 2"
   s title(41)="DC Bus 2,4 Energized"
   s title(81)="DC Load On"
   ! plotrun      ! standard plots
   ! print/noheader/file="run2.txt"/all
   action /clear
END !run2

PROCEDURE run3 ! IPS Motor Start: DC Bus 2,4 AC+DC Loads
   initrun    ! set generic initial conditions
   action /var=tic/val=.f./loc=cnntpwsp1       ! disconnect DC bus 1
   !turn on the system components, start motor at t = 0.0
   action /var=-4.9/val=.t./loc=opsscm4a       ! DC bus 4 converters on
   action /var=-4.9/val=.t./loc=opsscm4b
   action /var=-4.9/val=.t./loc=opres34a       ! DC bus 3/4 resistive load
   action /var=-4.7/val=.t./loc=opsscm2a       ! DC bus 2 converter on
   action /var=-4.5/val=.t./loc=opssim12a      ! ship service inverter on
   action /var=0.0/val=.t./loc=opmc34b     ! DC bus 3/4 induction motor

   !start the study
   s tstop=0.0, cint=1.0e-2, maxt(1)=5.0e-4
   start
   s tstop=10.0, cint=1.0e-2, maxt(1)=5.0e-4
   cont
   s title=120*" " ! clear title
   ! plot title 12345678901234567890123456789012345678901234567890 (40
characters/line)
   s title(01)="IPS Motor Start Run 3"
   s title(41)="DC Bus 2,4 AC Bus 3 Energized"
   s title(81)="DC Load, AC Load  On"
   ! plotrun      ! standard plots
   ! print/noheader/file="run3.txt"/all
   action /clear
END !run3

PROCEDURE run4 ! IPS Motor Start: Bus 1,2,4 AC+DC Loads
   initrun    ! set generic initial conditions
   action /var=tic/val=.f./loc=optran13    ! disconnect DC bus 1 -> 3
   !turn on the system components, start motor at t = 0.0
   action /var=-4.9/val=.t./loc=opsscm4a       ! DC bus 4 converters on
   action /var=-4.9/val=.t./loc=opsscm4b
   action /var=-4.9/val=.t./loc=opres34a       ! DC bus 3/4 resistive load
```

```
   action /var=-4.8/val=.t./loc=opsscm1a       ! DC bus 1 converter on

   action /var=-4.7/val=.t./loc=opsscm2a       ! DC bus 2 converter on
   action /var=-4.5/val=.t./loc=opssim12a      ! ship service inverter on
   action /var=0.0/val=.t./loc=opmc34b     ! DC bus 3/4 induction motor

   !start the study
   s tstop=0.0, cint=1.0e-2, maxt(1)=5.0e-4
   start
   s tstop=10.0, cint=1.0e-2, maxt(1)=5.0e-4
   cont
   s title=120*" " ! clear title
   ! plot title 12345678901234567890123456789012345678901234567890
   s title(01)="IPS Motor Start Run 4"
   s title(41)="DC Bus 1,2,4 AC Bus 3 Energized"
   s title(81)="DC Load, AC Load  On"
   ! plotrun      ! standard plots
   ! print/noheader/file="run4.txt"/all
   action /clear
END !run4

PROCEDURE run5 ! IPS Motor Start: Bus 1,2,3a,4 AC+DC Loads
   initrun    ! set generic initial conditions
   !turn on the system components, start motor at t = 0.0
   action /var=-4.9/val=.t./loc=opsscm4a       ! DC bus 4 converters on
   action /var=-4.9/val=.t./loc=opsscm4b
   action /var=-4.9/val=.t./loc=opres34a       ! DC bus 3/4 resistive load
   action /var=-4.8/val=.t./loc=opsscm1a       ! DC bus 1 converter on

   action /var=-4.7/val=.t./loc=opsscm2a       ! DC bus 2 converter on
   action /var=-4.8/val=.t./loc=opsscm3a       ! DC Bus 3a converter
   action /var=-4.5/val=.t./loc=opssim12a      ! ship service inverter on
   action /var=0.0/val=.t./loc=opmc34b     ! DC bus 3/4 induction motor
   !start the study
   s tstop=0.0, cint=1.0e-2, maxt(1)=5.0e-4
   start
   s tstop=10.0, cint=1.0e-2, maxt(1)=5.0e-4
   cont
   s title=120*" " ! clear title
   ! plot title 12345678901234567890123456789012345678901234567890
   s title(01)="IPS Motor Start Run 5"
   s title(41)="DC Bus 1,2,3a,4 AC Bus 3 Energized"
   s title(81)="DC Load, AC Load  On"
   ! plotrun      ! standard plots
   ! print/noheader/file="run5.txt"/all
   action /clear
END !run5

PROCEDURE run6 ! IPS Motor Start: Bus 1,2,3a,4 AC+DC Loads
   initrun    ! set generic initial conditions
   !turn on the system components, start motor at t = 0.0
   action /var=-4.9/val=.t./loc=opsscm4a       ! DC bus 4 converters on
   action /var=-4.9/val=.t./loc=opsscm4b
   action /var=-4.9/val=.t./loc=opres34a       ! DC bus 3/4 resistive load
```

```
   action /var=-4.8/val=.t./loc=opsscm1a       ! DC bus 1 converter on

   action /var=-4.7/val=.t./loc=opsscm2a       ! DC bus 2 converter on
   action /var=-4.8/val=.t./loc=opsscm3a       ! DC Bus 3a converters
   action /var=-4.8/val=.t./loc=opsscm3b
   action /var=-4.5/val=.t./loc=opssim12a      ! ship service inverter on
   action /var=0.0/val=.t./loc=opmc34b     ! DC bus 3/4 induction motor
   !start the study
   s tstop=0.0, cint=1.0e-2, maxt(1)=5.0e-4
   start
   s tstop=10.0, cint=1.0e-2, maxt(1)=5.0e-4
   cont
   s title=120*" " ! clear title
   ! plot title 12345678901234567890123456789012345678901234567890
   s title(01)="IPS Motor Start Run 6"
   s title(41)="DC Bus 1,2,3a,4 AC Bus 3 Energized"
   s title(81)="DC Load, AC Load  On"
   ! plotrun      ! standard plots
   ! print/noheader/file="run6.txt"/all
   action /clear
END !run6

PROCEDURE run7 ! IPS Motor Start: Bus 1,2,3a,3b AC+DC Loads
   initrun    ! set generic initial conditions
   action /var=tic/val=.f./loc=optran24    ! disconnect DC bus 2 -> 4
   !turn on the system components, start motor at t = 0.0
   action /var=-4.9/val=.t./loc=opsscm1a       ! DC bus 1 converter on

   action /var=-4.9/val=.t./loc=opsscm2a       ! DC bus 2 converter on
   action /var=-4.9/val=.t./loc=opsscm3a       ! DC Bus 3 converters on
   action /var=-4.9/val=.t./loc=opsscm3b
   action /var=-4.9/val=.t./loc=opres34a       ! DC bus 3/4 resistive load
   action /var=-4.5/val=.t./loc=opssim12a      ! ship service inverter on
   action /var=0.0/val=.t./loc=opmc34b     ! DC bus 3/4 induction motor

   !start the study
   s tstop=0.0, cint=1.0e-2, maxt(1)=5.0e-4
   start
   s tstop=10.0, cint=1.0e-2, maxt(1)=5.0e-4
   cont
   s title=120*" " ! clear title
   ! plot title 12345678901234567890123456789012345678901234567890
   s title(01)="IPS Motor Start Run 7"
   s title(41)="DC Bus 1,2,3a,3b AC Bus 3 Energized"
   s title(81)="DC Load, AC Load  On"
   ! plotrun      ! standard plots
   ! print/noheader/file="run7.txt"/all
   action /clear
END !run7

PROCEDURE run8 ! IPS Motor Start: Bus 1,3 AC+DC Loads
   initrun    ! set generic initial conditions
   action /var=tic/val=.f./loc=cnntpwsp2       ! disconnect DC bus 2
   action /var=tic/val=.f./loc=optran24    ! disconnect DC bus 2 -> 4
   !turn on the system components, start motor at t = 0.0
```

```
   action /var=-4.9/val=.t./loc=opsscm1a       ! DC bus 1 converter on

   action /var=-4.9/val=.t./loc=opsscm3a       ! DC Bus 3 converters on
   action /var=-4.9/val=.t./loc=opsscm3b
   action /var=-4.9/val=.t./loc=opres34a       ! DC bus 3/4 resistive load
   action /var=-4.5/val=.t./loc=opssim12a      ! ship service inverter on
   action /var=0.0/val=.t./loc=opmc34b     ! DC bus 3/4 induction motor

   !start the study
   s tstop=0.0, cint=1.0e-2, maxt(1)=5.0e-4
   start
   s tstop=10.0, cint=1.0e-2, maxt(1)=5.0e-4
   cont
   s title=120*" " ! clear title
   ! plot title 12345678901234567890123456789012345678901234567890
   s title(01)="IPS Motor Start Run 8"
   s title(41)="DC Bus 1,3 AC Bus 3 Energized"
   s title(81)="DC Load, AC Load  On"
   ! plotrun      ! standard plots
   ! print/noheader/file="run8.txt"/all
   action /clear
END !run8

PROCEDURE test1init
s nrwitg=.t.
run1
s psfspl=0.9
s devplt=5
file/pltfile='run1init.ps'
plotinit
plot/close
s nrwitg=.f.
END ! test1init

PROCEDURE test1
s nrwitg=.f.
run1
s psfspl=0.9
s devplt=5
file/pltfile='run1.ps'
plotrun
plot/close
print/noheader/file="run1.txt"/all
s nrwitg=.f.
END ! test1init

PROCEDURE test2init
s nrwitg=.t.
run2
s psfspl=0.9
s devplt=5
file/pltfile='run2init.ps'
plotinit
plot/close
s nrwitg=.f.
```

```
END ! test2init

PROCEDURE test2
s nrwitg=.f.
run2
s psfspl=0.9
s devplt=5
file/pltfile='run2.ps'
plotrun
plot/close
print/noheader/file="run2.txt"/all
s nrwitg=.f.
END ! test2init

PROCEDURE test3init
s nrwitg=.t.
run3
s psfspl=0.9
s devplt=5
file/pltfile='run3init.ps'
plotinit
plot/close
s nrwitg=.f.
END ! test3init

PROCEDURE test3
s nrwitg=.f.
run3
s psfspl=0.9
s devplt=5
file/pltfile='run3.ps'
plotrun
plot/close
print/noheader/file="run3.txt"/all
s nrwitg=.f.
END ! test3init

PROCEDURE test4init
s nrwitg=.t.
run4
s psfspl=0.9
s devplt=5
file/pltfile='run4init.ps'
plotinit
plot/close
s nrwitg=.f.
END ! test4init

PROCEDURE test4
s nrwitg=.f.
run4
s psfspl=0.9
s devplt=5
file/pltfile='run4.ps'
plotrun
```

```
plot/close
print/noheader/file="run4.txt"/all
s nrwitg=.f.
END ! test4init


PROCEDURE test5init
s nrwitg=.t.
run5
s psfspl=0.9
s devplt=5
file/pltfile='run5init.ps'
plotinit
plot/close
s nrwitg=.f.
END ! test5init

PROCEDURE test5
s nrwitg=.f.
run5
s psfspl=0.9
s devplt=5
file/pltfile='run5.ps'
plotrun
plot/close
print/noheader/file="run5.txt"/all
s nrwitg=.f.
END ! test5init

PROCEDURE test6init
s nrwitg=.t.
run6
s psfspl=0.9
s devplt=5
file/pltfile='run6init.ps'
plotinit
plot/close
s nrwitg=.f.
END ! test6init

PROCEDURE test6
s nrwitg=.f.
run6
s psfspl=0.9
s devplt=5
file/pltfile='run6.ps'
plotrun
plot/close
print/noheader/file="run6.txt"/all
s nrwitg=.f.

END ! test7init
PROCEDURE test7init
s nrwitg=.t.
run7
```

```
s psfspl=0.9
s devplt=5
file/pltfile='run7init.ps'
plotinit
plot/close
s nrwitg=.f.
END ! test7init

PROCEDURE test7
s nrwitg=.f.
run7
s psfspl=0.9
s devplt=5
file/pltfile='run7.ps'
plotrun
plot/close
print/noheader/file="run7.txt"/all
s nrwitg=.f.
END ! test7init

PROCEDURE test8init
s nrwitg=.t.
run8
s psfspl=0.9
s devplt=5
file/pltfile='run8init.ps'
plotinit
plot/close
s nrwitg=.f.
END ! test8init

PROCEDURE test8
s nrwitg=.f.
run8
s psfspl=0.9
s devplt=5
file/pltfile='run8.ps'
plotrun
plot/close
print/noheader/file="run8.txt"/all
s nrwitg=.f.
END ! test8init
```

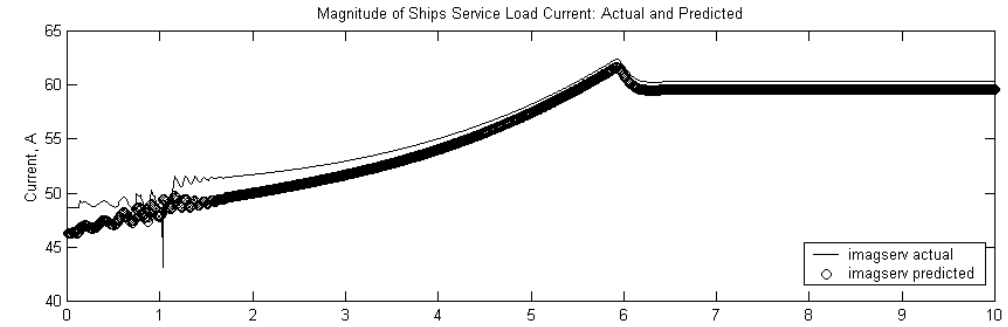
12.4 Appendix D – Filter for Data Compatability

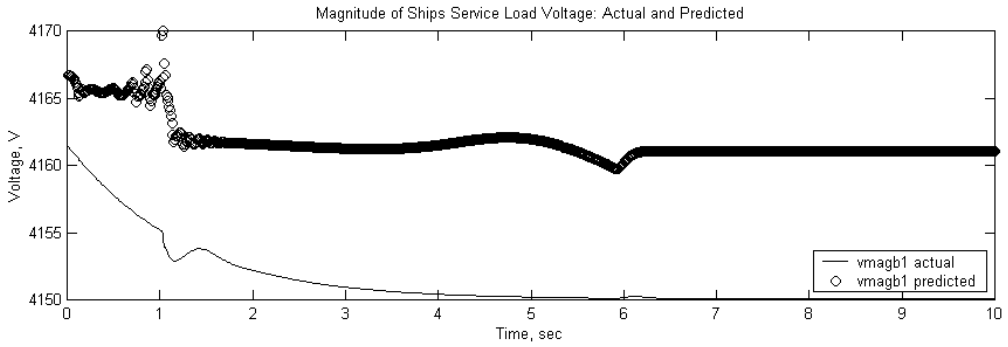
```
% filter for 1/C Cerrito neural net project
% E. Zivi    4/1/00

i=input('Enter run # : ');
varname=['run' num2str(i)];
infilename=[varname '.txt'];
outfilename=[varname 'f.nna'];
load(infilename);
cutx=eval([varname '(:,6:25)']);
[r,c]=size(cutx);
shiftx=[cutx(1:r-1,:) cutx(2:r,:) cutx(1:r-1,:)];
fid=fopen(outfilename,'w');
for i=1:r-1,
   fprintf(fid,'%12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f
%12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f
%12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f
%12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f
%12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f
%12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f
%12.4f %12.4f\n',shiftx(i,:));
end
fclose(fid);
```

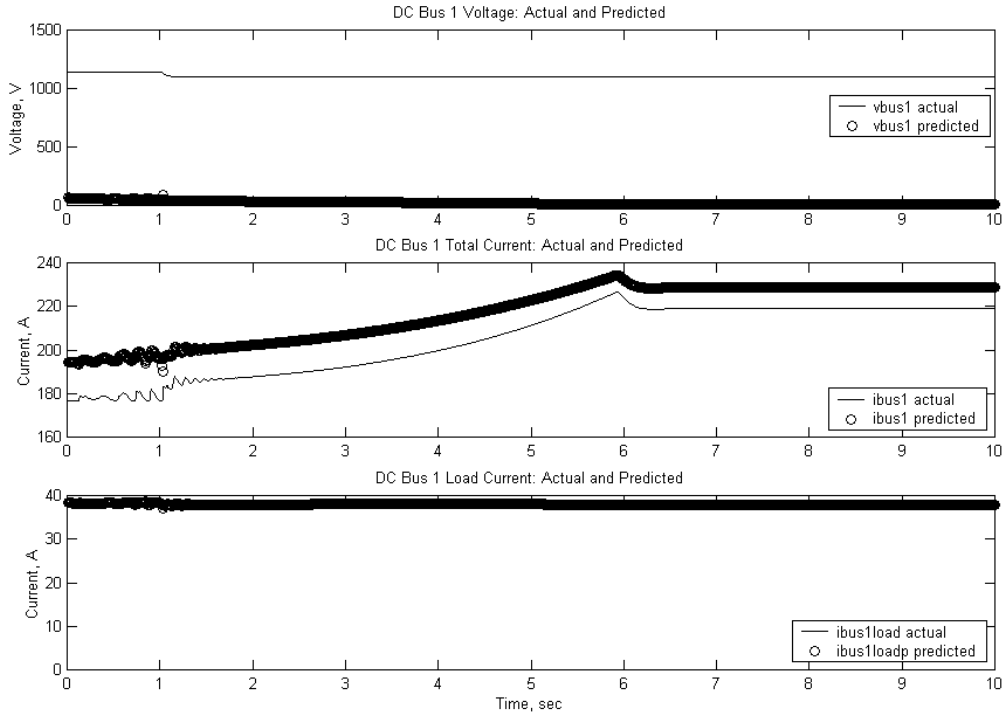
12.5 Appendix E – Fault Data Filter

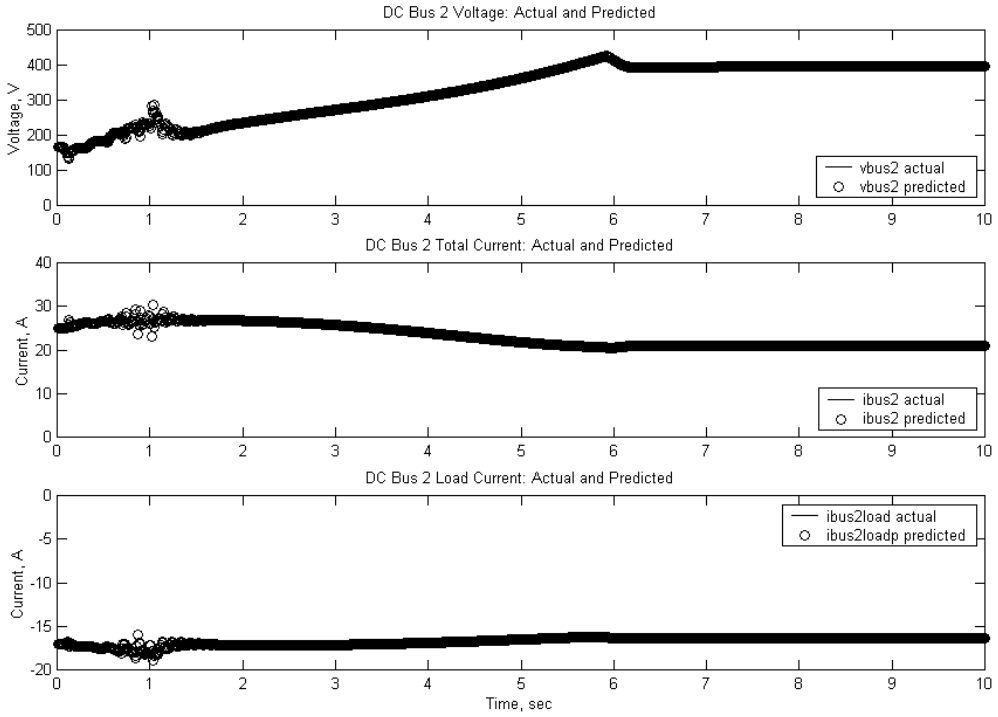
```
% Fault Data Filter for neural net project
% J. Cerrito    4/7/00

i=input('Enter run # : ');
varname=['run' num2str(i)];
infilename=[varname '.txt'];
outfilename=[varname 'falt.nna'];
load(infilename);
matrixf=eval([varname '(:,:)']);
[r,c]=size(matrixf);

for i=1:r,
   for j=1:c-20,
      sign=rand(1);
      if (sign<.6 & sign>.4)
      matrixf(i,j)=matrixf(i,j);
      end
      if sign>=.6
      matrixf(i,j)=((.1*rand(1)*matrixf(i,j))+matrixf(i,j));
      end
      if sign<.4
      matrixf(i,j)=(matrixf(i,j)-(.1*rand(1)*matrixf(i,j)));
      end
   end
end
fid=fopen(outfilename,'w');
for i=1:r-1,
   fprintf(fid,'%12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f
%12.4f %12.4f%12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f
%12.4f %12.4f%12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f
%12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f
%12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f
%12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f
%12.4f %12.4f\n',matrixf(i,:));
end
fclose(fid);
```

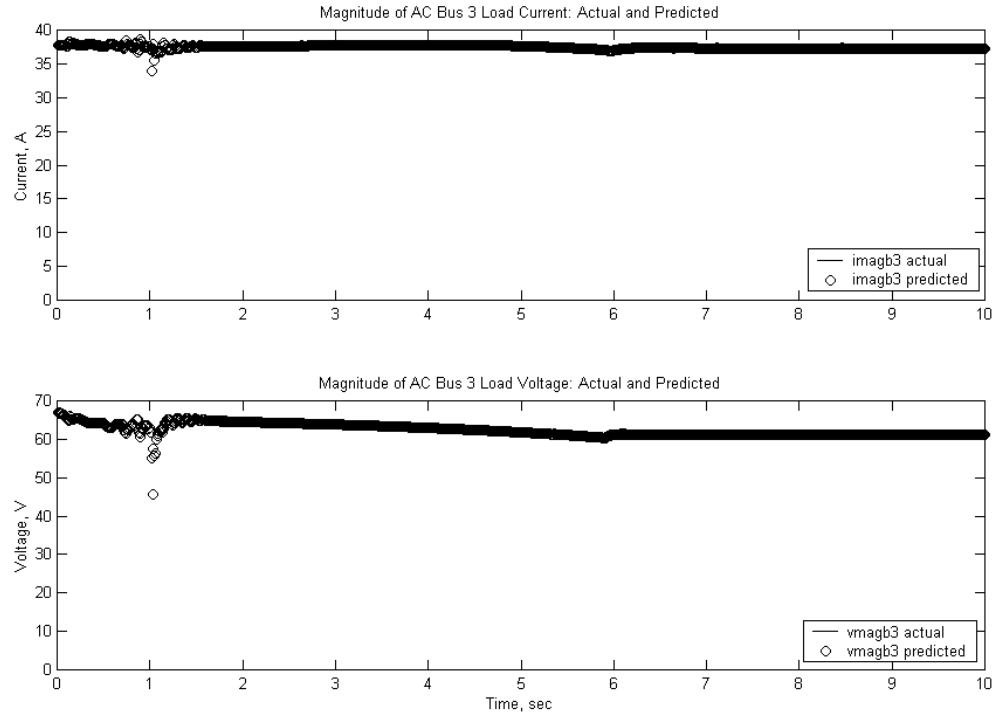
12.6 Appendix F – 40-20 Variable Structure Neural Network: Run 1

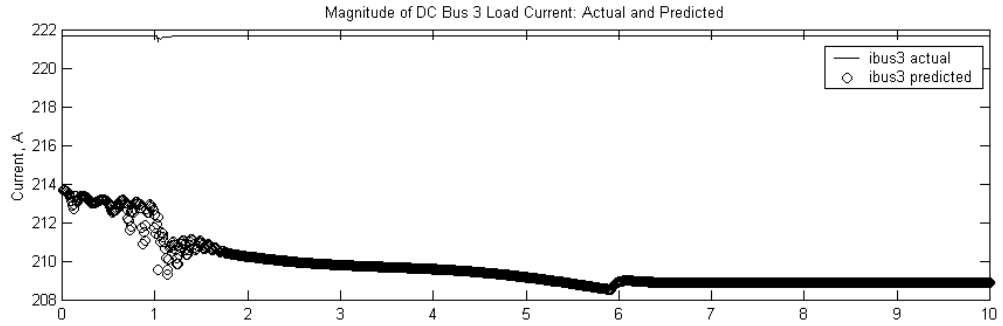


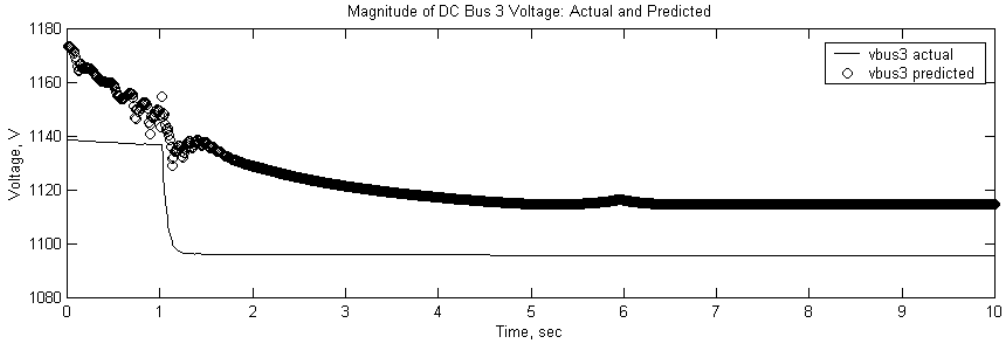


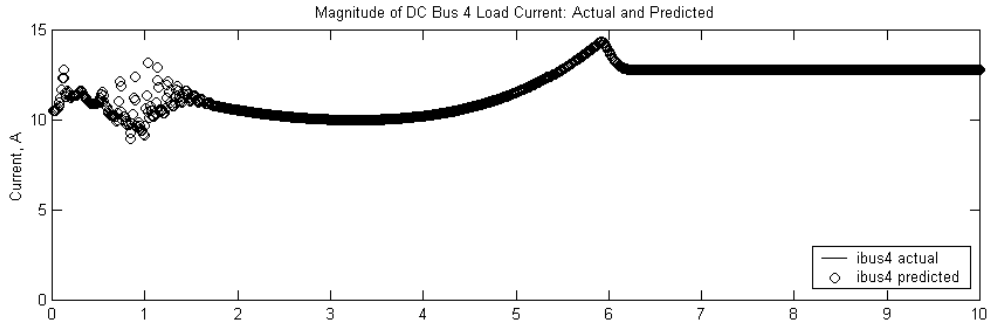


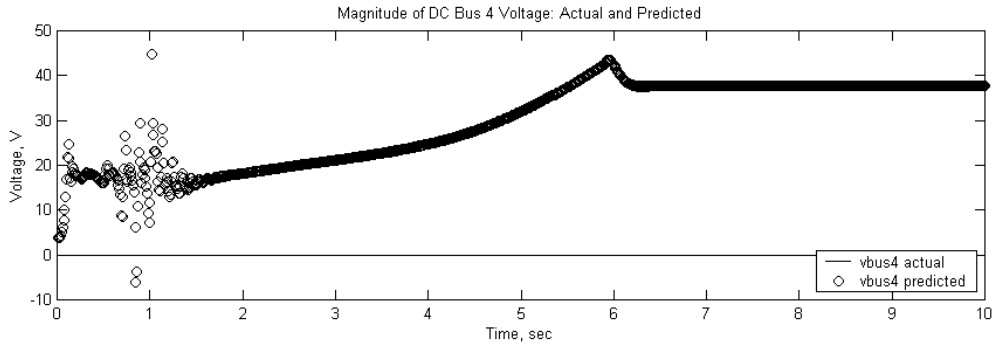

DC Bus 2 Voltage: Actual and Predicted
Voltage, V
vbus2 actual
vbus2 predicted
DC Bus 2 Total Current: Actual and Predicted
Current, A
ibus2 actual
ibus2 predicted
DC Bus 2 Load Current: Actual and Predicted
Current, A
ibus2load actual
ibus2loadp predicted
Time, sec

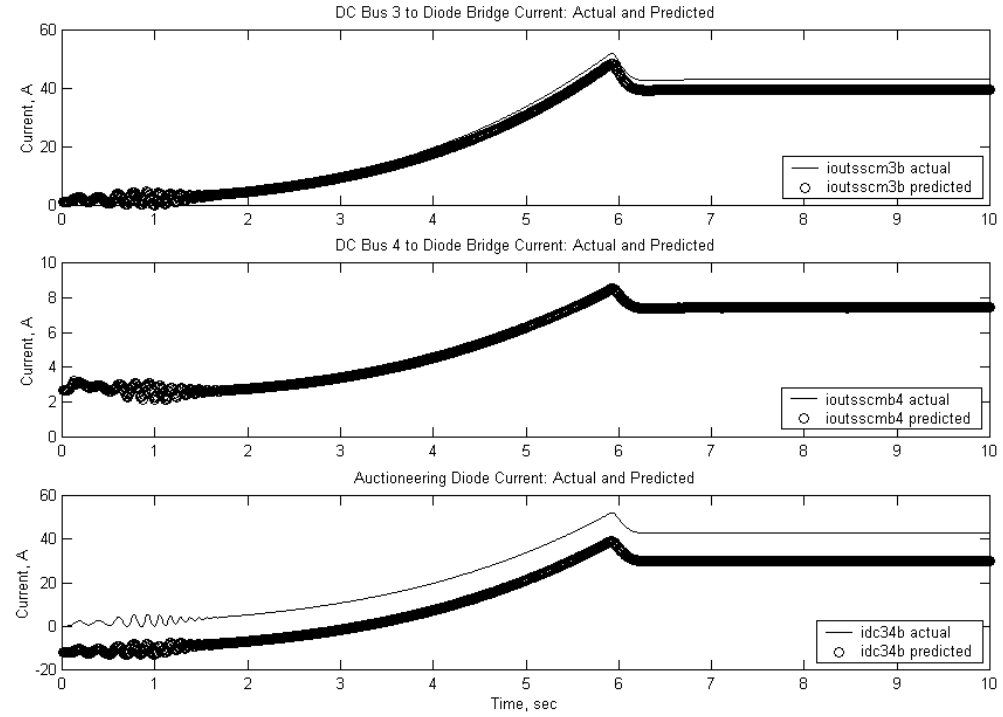

Magnitude of AC Bus 3 Load Current: Actual and Predicted
Current, A
imagb3 actual
imagb3 predicted
Magnitude of AC Bus 3 Load Voltage: Actual and Predicted
Voltage, V
vmagb3 actual
vmagb3 predicted
Time, sec

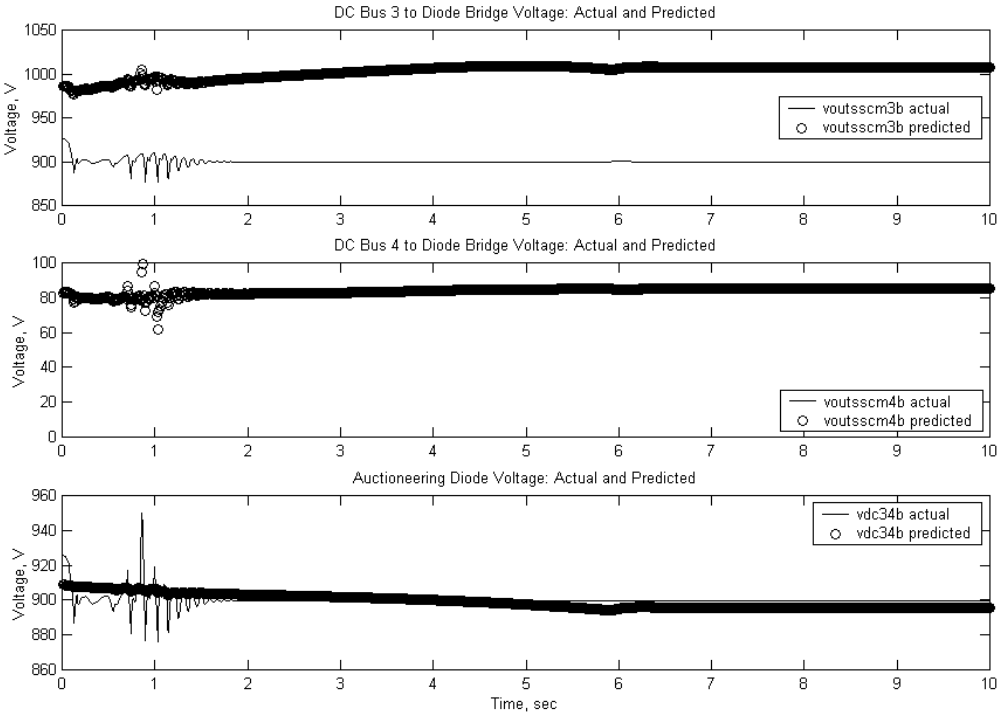

Magnitude of DC Bus 3 Load Current: Actual and Predicted
Current, A
ibus3 actual
ibus3 predicted


Magnitude of DC Bus 3 Voltage: Actual and Predicted
Voltage, V
Time, sec
vbus3 actual
vbus3 predicted

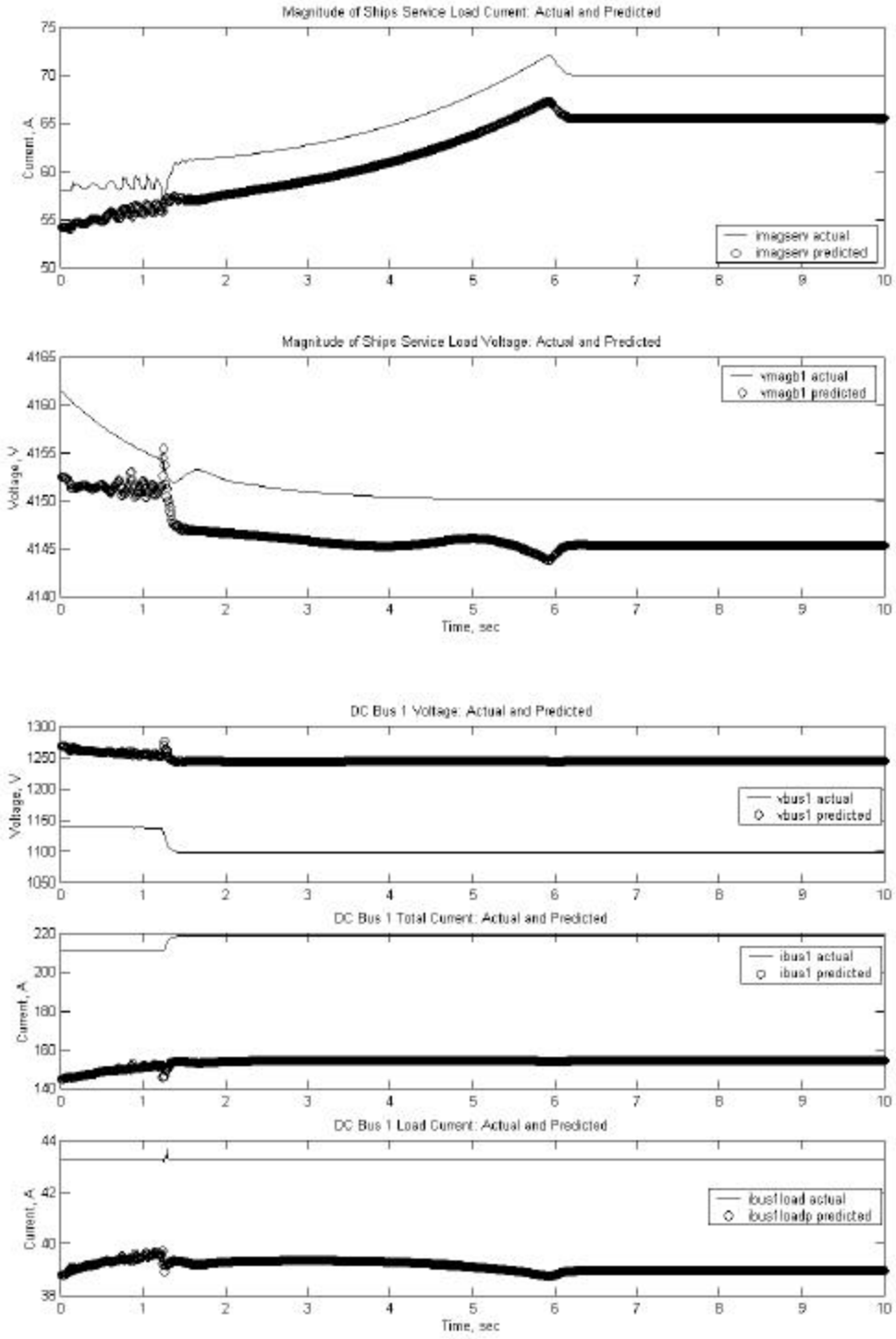

Magnitude of DC Bus 4 Load Current: Actual and Predicted
Current, A
ibus4 actual
ibus4 predicted

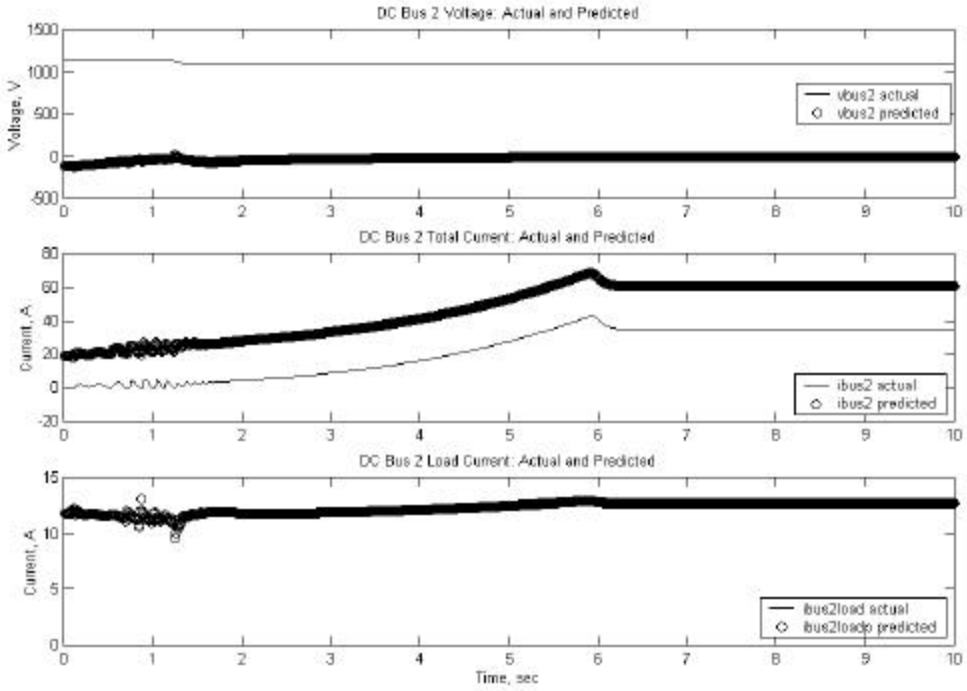

Magnitude of DC Bus 4 Voltage: Actual and Predicted
Voltage, V
Time, sec
vbus4 actual
vbus4 predicted

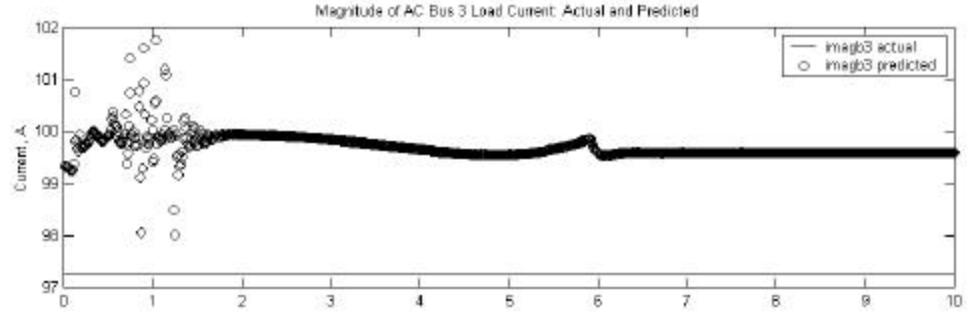

DC Bus 3 to Diode Bridge Current: Actual and Predicted
Current, A
ioutsscm3b actual
ioutsscm3b predicted
DC Bus 4 to Diode Bridge Current: Actual and Predicted
Current, A
ioutsscmb4 actual
ioutsscmb4 predicted
Auctioneering Diode Current: Actual and Predicted
Current, A
idc34b actual
idc34b predicted
Time, sec

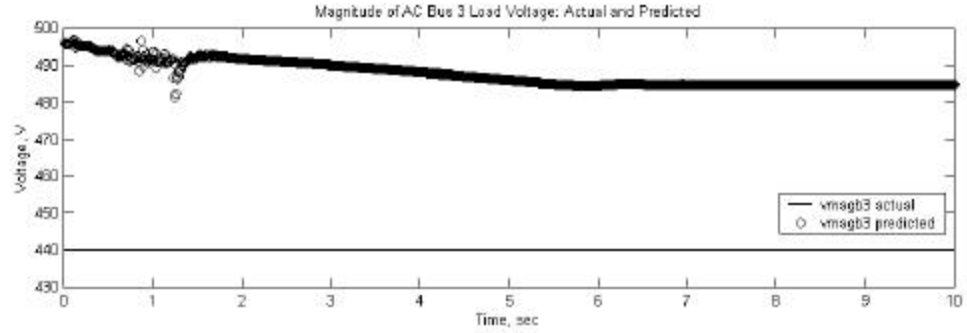

DC Bus 3 to Diode Bridge Voltage: Actual and Predicted
Voltage, V
voutsscm3b actual
voutsscm3b predicted
DC Bus 4 to Diode Bridge Voltage: Actual and Predicted
Voltage, V
voutsscm4b actual
voutsscm4b predicted
Auctioneering Diode Voltage: Actual and Predicted
Voltage, V
vdc34b actual
vdc34b predicted
Time, sec

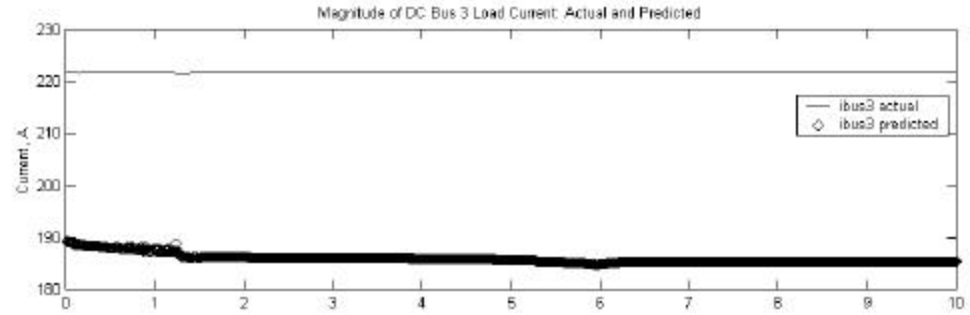
12.7 Appendix G – 40-20 Variable Structure Neural Network: Run 5

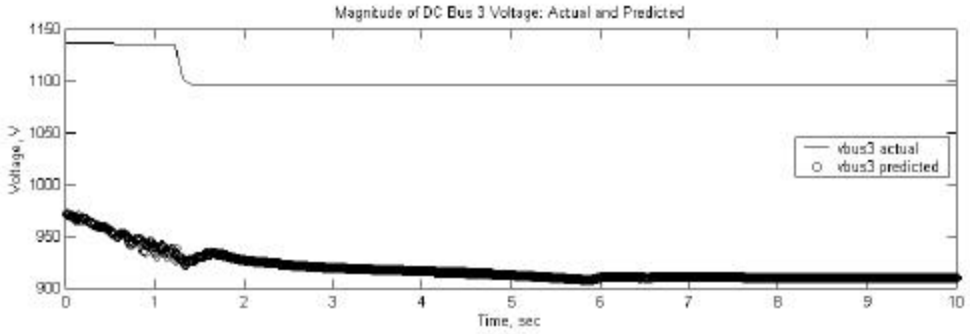


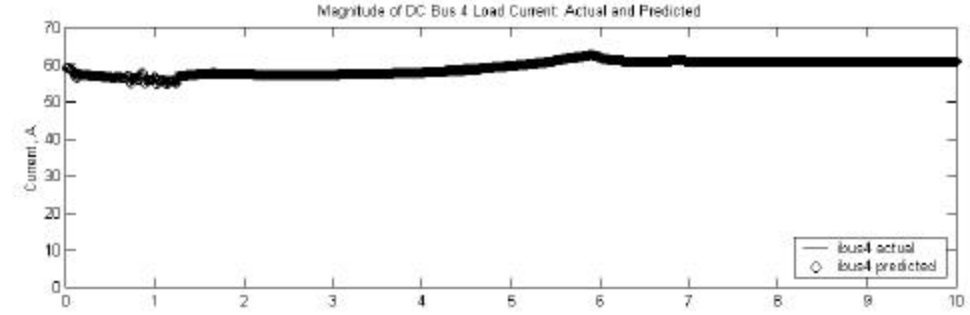

DC Bus 2 Voltage: Actual and Predicted
Voltage, V
vbus2 actual
vbus2 predicted
DC Bus 2 Total Current: Actual and Predicted
Current, A
ibus2 actual
ibus2 predicted
DC Bus 2 Load Current: Actual and Predicted
Current, A
ibus2load actual
ibus2loadp predicted
Time, sec

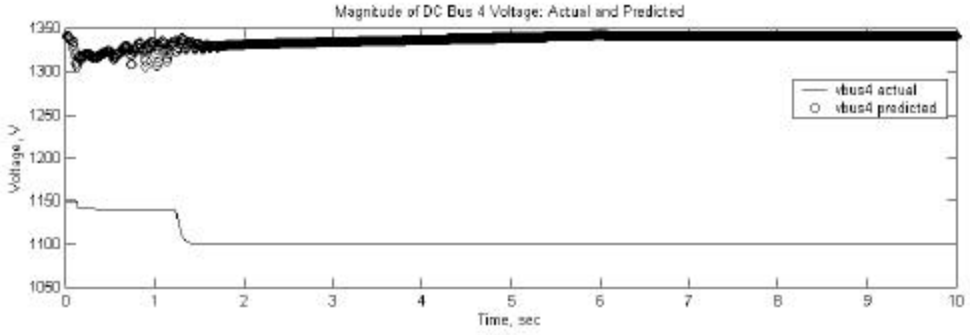

Magnitude of AC Bus 3 Load Current: Actual and Predicted
Current, A
imagb3 actual
imagb3 predicted

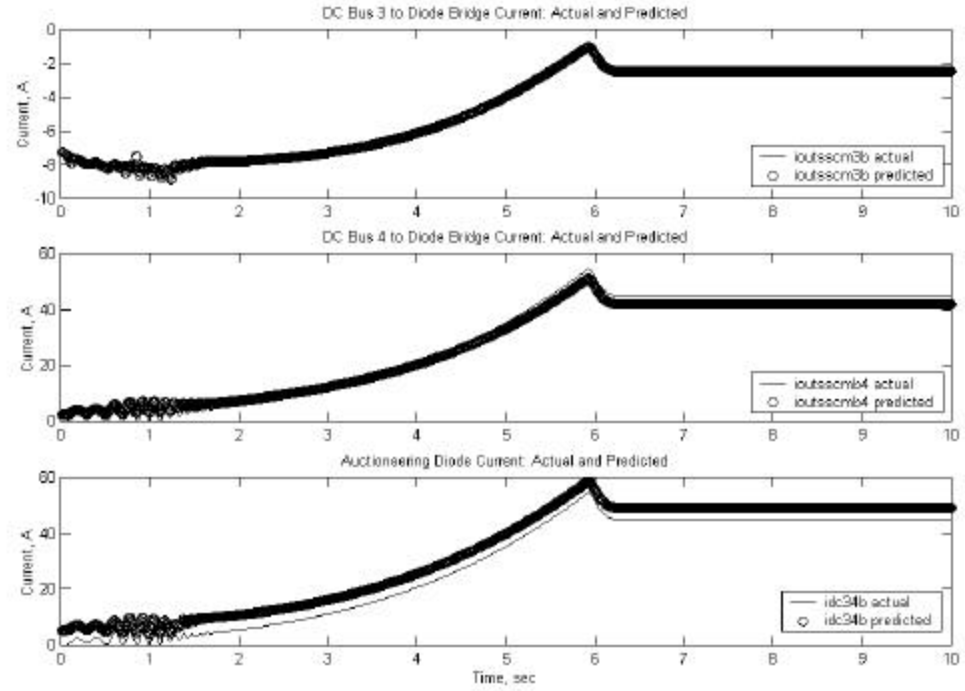

Magnitude of AC Bus 3 Load Voltage: Actual and Predicted
Voltage, V
vmagb3 actual
vmagb3 predicted
Time, sec

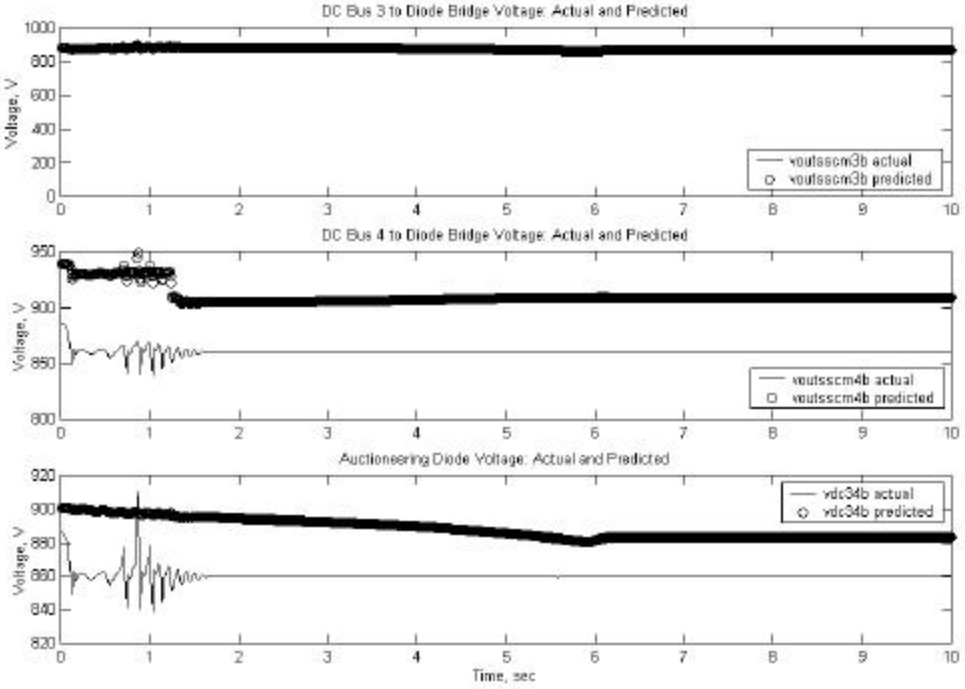

Magnitude of DC Bus 3 Load Current: Actual and Predicted
Current, A
ibus3 actual
ibus3 predicted


Magnitude of DC Bus 3 Voltage: Actual and Predicted
Voltage, V
vbus3 actual
vbus3 predicted
Time, sec

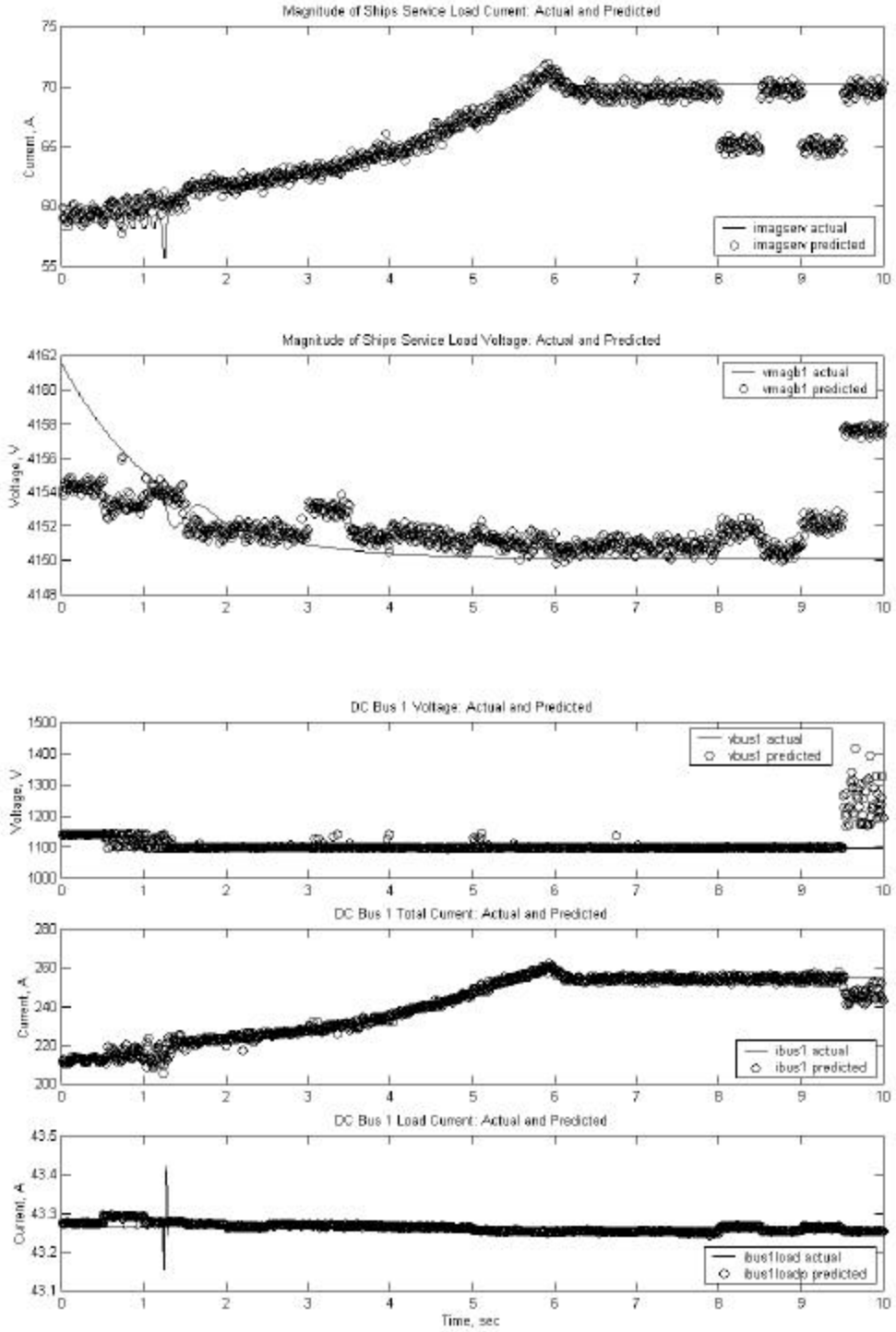

Magnitude of DC Bus 4 Load Current: Actual and Predicted
Current, A
ibus4 actual
ibus4 predicted

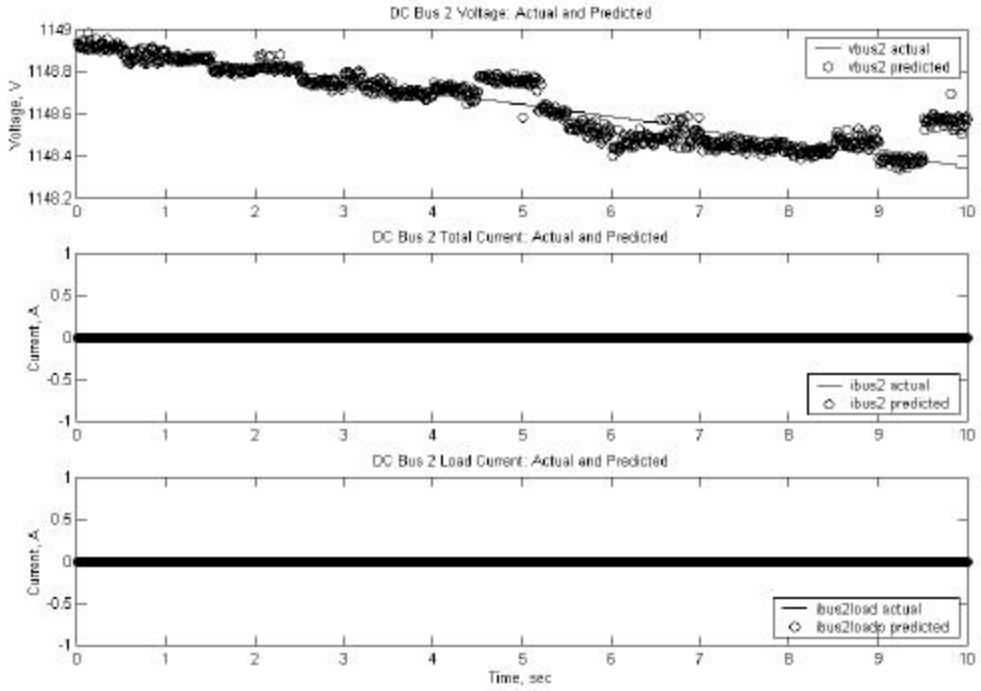

Magnitude of DC Bus 4 Voltage: Actual and Predicted
Voltage, V
vbus4 actual
vbus4 predicted
Time, sec

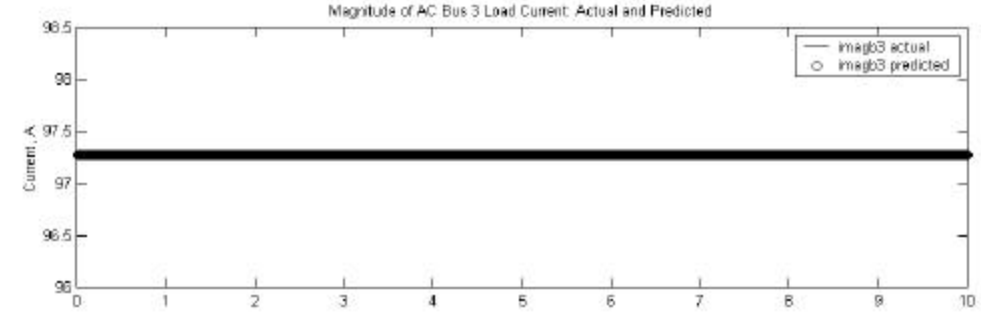

DC Bus 3 to Diode Bridge Current: Actual and Predicted
Current, A
ioutsscm3b actual
ioutsscm3b predicted
DC Bus 4 to Diode Bridge Current: Actual and Predicted
Current, A
ioutsscmb4 actual
ioutsscmb4 predicted
Auctioneering Diode Current: Actual and Predicted
Current, A
idc34b actual
idc34b predicted
Time, sec

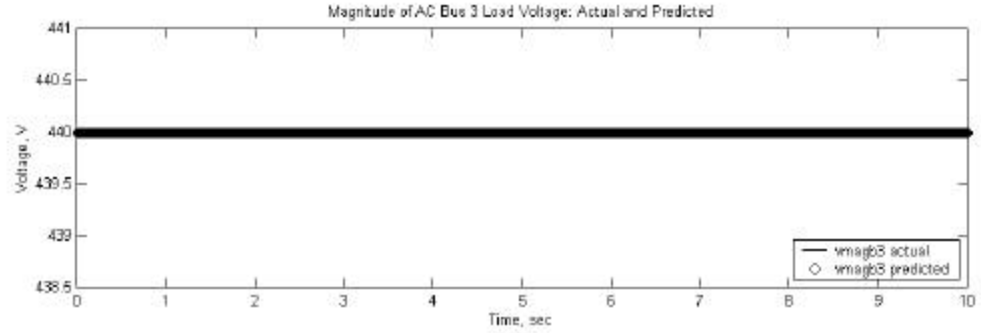

DC Bus 3 to Diode Bridge Voltage: Actual and Predicted
Voltage, V
voutsscm3b actual
voutsscm3b predicted
DC Bus 4 to Diode Bridge Voltage: Actual and Predicted
Voltage, V
voutsscm4b actual
voutsscm4b predicted
Auctioneering Diode Voltage: Actual and Predicted
Voltage, V
vdc34b actual
vdc34b predicted
Time, sec

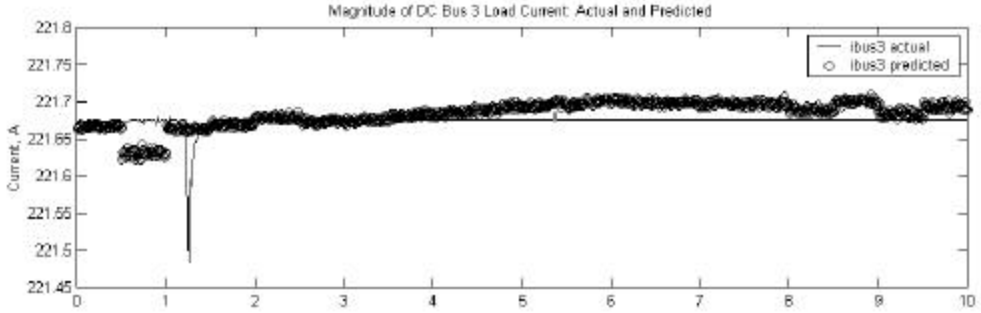
12.8 Appendix H – 40-20 Faulted Neural Network: Run 6 Corrupted Data

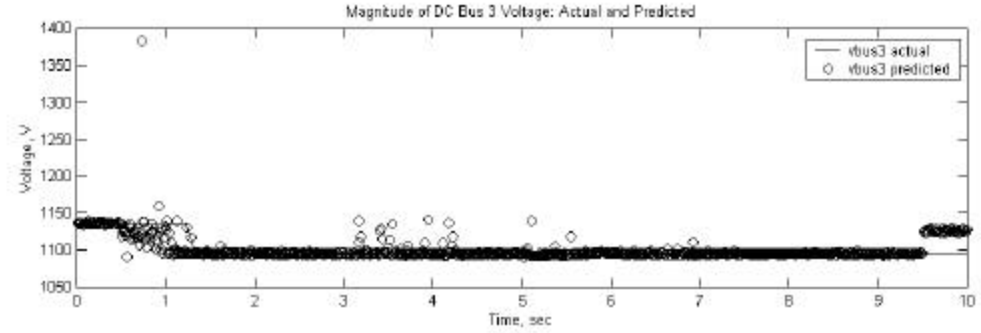


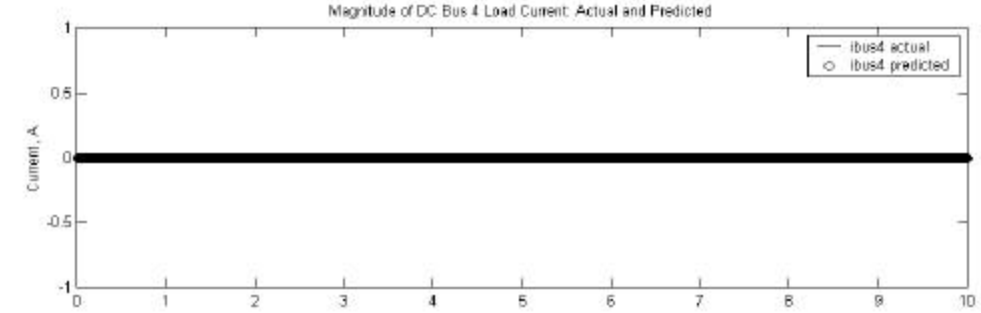

DC Bus 2 Voltage: Actual and Predicted
Voltage, V
vbus2 actual
vbus2 predicted
DC Bus 2 Total Current: Actual and Predicted
Current, A
ibus2 actual
ibus2 predicted
DC Bus 2 Load Current: Actual and Predicted
Current, A
ibus2load actual
ibus2loadp predicted
Time, sec

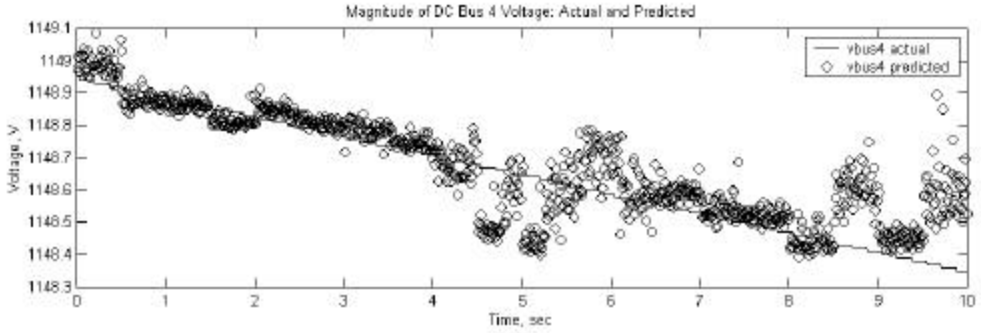

Magnitude of AC Bus 3 Load Current: Actual and Predicted
Current, A
imagb3 actual
imagb3 predicted

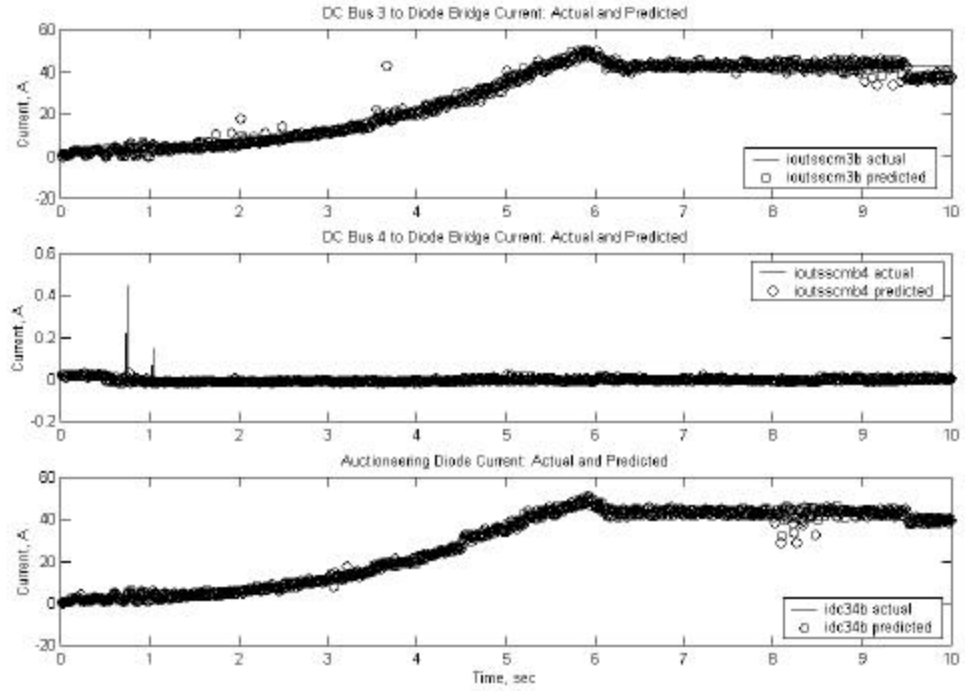

Magnitude of AC Bus 3 Load Voltage: Actual and Predicted
Voltage, V
vmagb3 actual
vmagb3 predicted
Time, sec

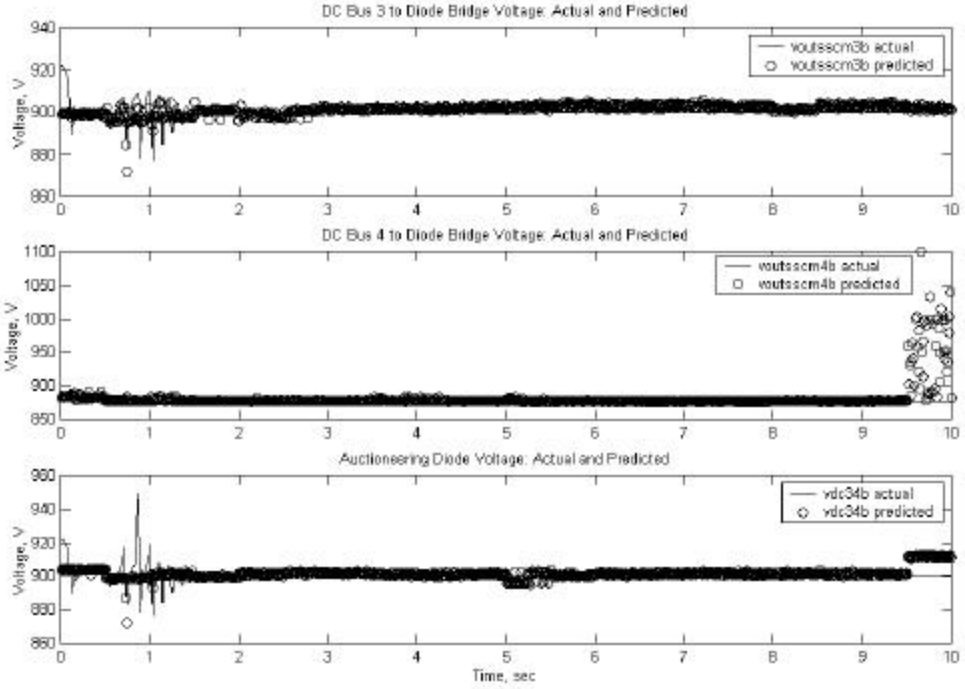

Magnitude of DC Bus 3 Load Current: Actual and Predicted
ibus3 actual
ibus3 predicted
Current, A


Magnitude of DC Bus 3 Voltage: Actual and Predicted
vbus3 actual
vbus3 predicted
Voltage, V
Time, sec


Magnitude of DC Bus 4 Load Current: Actual and Predicted
ibus4 actual
ibus4 predicted
Current, A


Magnitude of DC Bus 4 Voltage: Actual and Predicted
vbus4 actual
vbus4 predicted
Voltage, V
Time, sec


DC Bus 3 to Diode Bridge Current: Actual and Predicted
Current, A
ioutsscm3b actual
ioutsscm3b predicted
DC Bus 4 to Diode Bridge Current: Actual and Predicted
Current, A
ioutsscm4 actual
ioutsscm4 predicted
Auctioneering Diode Current: Actual and Predicted
Current, A
idc34b actual
idc34b predicted
Time, sec


DC Bus 3 to Diode Bridge Voltage: Actual and Predicted
Voltage, V
voutsscm3b actual
voutsscm3b predicted
DC Bus 4 to Diode Bridge Voltage: Actual and Predicted
Voltage, V
voutsscm4b actual
voutsscm4b predicted
Auctioneering Diode Voltage: Actual and Predicted
Voltage, V
vdc34b actual
vdc34b predicted
Time, sec